Ari Saptawijaya

Master in Computer Science

Machine Ethics via Logic Programming

Dissertação para obtenção do Grau de
Doutor em Informática

Orientador: Luís Moniz Pereira, Professor Catedrático,
Universidade Nova de Lisboa

Júri:

Presidente:	Prof. José Augusto Legatheaux Martins
Arguentes:	Assistant Prof. Alexandre Miguel dos Santos Martins Pinto
	Dr. Terrance Lee Swift
Vogais:	Associate Prof. João Alexandre Carvalho Pinheiro Leite
	Senior Lecturer, Dr. Fariba Sadri

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

July, 2015

Machine Ethics via Logic Programming

For Ananda

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my supervisor, Prof. Luís Moniz Pereira, who has introduced me to this challenging topic and provided me the guidance in exploring this terra incognita, whilst also giving me the freedom to explore it on my own. I am really grateful to have Luís as my supervisor for his willingness to help at any time. Luís always encouraged me, with his patience, to go through my difficult situations, which kept me on track, and to eventually finish this thesis. I have learned a lot from his abundant knowledge in this interdisciplinary field (including, of course, his expertise in Logic Programming), absorbed his endless original ideas, and enjoyed many good times, where he shared his adventurous experiences. One is not always so lucky to have such a friendly and helpful supervisor.

I am thankful to Dr. João Alexandre Leite, Dr. Fariba Sadri, and Dr. Terrance Swift as the members of my thesis advisory committee, for their suggestions and advices during the discussion of my thesis proposal. I also thank them and Dr. Alexandre Miguel Pinto that have accepted to be the jury members of my thesis defence.

I am grateful to Prof. David S. Warren and Dr. Terrance Swift for their expertise in helping me with XSB Prolog, as well as for their specific suggestions in the development of tabling in abduction (TABDUAL) and updating (EVOLP/R) . I am also grateful to have an opportunity to discuss with Prof. Robert Kowalski about the idea of contextual abduction and rule name fluent mechanism in EVOLP/R. I would like to acknowledge Emmanuelle Dietz for our discussion on counterfactuals and other topics during her visit to Portugal, Gonçalo Lopes for explaining the idea of and how to use ACORDA for the work in this thesis, and Dr. Miguel Calejo for a number of suggestions to improve QUALM in future.

Prof. Nuno Correia, as the coordinator of the PhD Program in Computer Science, and the secretariat of the Department of Informatics (DI/FCT/UNL), particularly Mrs. Sandra Rainha and Mrs. Susana Pereira, have provided me with their responsive and kind assistance to deal with bureaucratic stuffs, and for all that I am greatly thankful to them.

I am indebted to Han The Anh and Martin Slota for their generous help during my first days in Portugal. The Anh has provided me with essential and practical information about doing a doctoral study in Portugal, and has kindly hosted me for several days in his shared apartment with Martin as soon as I arrived in Portugal. It is my pleasure to have a joint work with him on probabilistic moral reasoning, which becomes part of this thesis.

Davide D'Alimonte and Tamito Kajiyama (and his lovely family) have particularly spent good times with me and my family with fun activities together, and I greatly value

our cheerful friendship. Our joint research work in ocean color remote sensing was not less fun either.

I am also grateful to have an opportunity, in various occasions, to establish warm friendship with Luís Mota Ferreira & Maria Freitas Casimiro, Jan Cederquist & Ana Almeida Matos, and several people in the department during my doctoral study: Ankica Barišić, Jorge Costa, Sinan Egilmez, Sofia Gomes, Ricardo Gonçalves, Cédric Grueau, Matthias Knorr, Fábio Madeira, João Moura, Nguyen Van Hau, and Filipa Peleja. I have also been blessed for knowing an Indonesian community in Lisbon, with whom I never felt being far away from my homeland.

I acknowledge the support from Fundação para a Ciência e a Tecnologia (FCT) that has funded my doctoral study through grant SFRH/BD/72795/2010, as well as CENTRIA and DI/FCT/UNL for the supplementary funding that allowed me to attend a number of conferences. I am also grateful to the Faculty of Computer Science at my home university, Universitas Indonesia, for permitting me to take a leave of absence to pursue my doctoral study.

The completion of my study would have been impossible without the love and patience of my family, to whom I would like to express my heart-felt gratitude. My parents have always been supportive of me, even though from a long distance, with their unconditional love, all these years. Last but not least, my wife – Meiriana – and my (now not so little anymore) son – Ananda – have been the constant source of encouragement and happiness throughout this endeavour. To all of you, this thesis is dedicated.

Abstract

Machine ethics is an interdisciplinary field of inquiry that emerges from the need of imbuing autonomous agents with the capacity of moral decision-making. While some approaches provide implementations in Logic Programming (LP) systems, they have not exploited LP-based reasoning features that appear essential for moral reasoning.

This PhD thesis aims at investigating further the appropriateness of LP, notably a combination of LP-based reasoning features, including techniques available in LP systems, to machine ethics. Moral facets, as studied in moral philosophy and psychology, that are amenable to computational modeling are identified, and mapped to appropriate LP concepts for representing and reasoning about them.

The main contributions of the thesis are twofold.

First, novel approaches are proposed for employing tabling in contextual abduction and updating – individually and combined – plus a LP approach of counterfactual reasoning; the latter being implemented on top of the aforementioned combined abduction and updating technique with tabling. They are all important to model various issues of the aforementioned moral facets.

Second, a variety of LP-based reasoning features are applied to model the identified moral facets, through moral examples taken off-the-shelf from the morality literature. These applications include: (1) Modeling moral permissibility according to the Doctrines of Double Effect (DDE) and Triple Effect (DTE), demonstrating deontological and utilitarian judgments via integrity constraints (in abduction) and preferences over abductive scenarios; (2) Modeling moral reasoning under uncertainty of actions, via abduction and probabilistic LP; (3) Modeling moral updating (that allows other – possibly overriding – moral rules to be adopted by an agent, on top of those it currently follows) via the integration of tabling in contextual abduction and updating; and (4) Modeling moral permissibility and its justification via counterfactuals, where counterfactuals are used for formulating DDE.

Keywords: abduction, counterfactual, logic programming, machine ethics, morality, non-monotonic reasoning, tabling.

RESUMO

A ética da máquina é um campo interdisciplinar de investigação que emerge da necessidade de embeber os agentes autónomos com a capacidade de decisão moral. Embora certas abordagens providenciem implementações que usam sistemas de Programação em Lógica (PL), elas não têm explorado as facetas do raciocínio baseado em PL que parecem ser essenciais ao raciocínio moral.

Esta tese de doutoramento almeja investigar mais longe a pertinência da PL para a ética da máquina, em especial por uma combinação de características para raciocínio da PL, inclusive técnicas já disponíveis em sistemas de PL. As facetas morais, tal como são estudadas na filosofia moral e na psicologia, são por nós identificadas, e mapeadas nos conceitos de PL apropriados para as representar e raciocinar sobre elas.

As principais contribuições desta tese são de dois tipos.

Em primeiro lugar, são propostas abordagens novas para o emprego de tabulação na abdução contextual e na actualização – quer individualmente quer em combinação – e ainda uma abordagem em PL ao raciocínio contrafactual; esta última é implementada sobre a supramencionada técnica de tabulação combinando a abdução e a actualização. Todas estas abordagens são importantes para modelar várias questões das já referidas facetas morais.

Em segundo lugar, diversas características do raciocínio em PL são utilizadas para modelar as facetas morais, através de exemplos de prateleira da literatura da moralidade. Tais aplicações incluem: (1) Modelação da permissibilidade segundo as Doutrinas do Duplo Efeito (DDE) e Triplo Efeito (DTE), demonstrando os juízos deontológicos e utilitários por via de condições de integridade (na abdução) e de preferências sobre os cenários abdutivos; (2) Modelação do raciocínio moral sujeito a incerteza de acções, via abdução e PL probabilística; (3) Modelação da actualização moral (permitindo que outras regras morais – possivelmente prevalecentes – possam ser adoptadas pelo agente, sobrepondo-se àquelas que presentemente segue) via a integração tabulativa da abdução contextual e da actualização; e (4) Modelação da permissibilidade moral e sua justificação via contrafactuais, em que estes são utilizados para formular a DDE.

Palavras-chave: abdução, contrafactual, programação em lógica, ética da máquina, raciocínio não-monótono, tabulação.

CONTENTS

List of Figures

List of Tables

GLOSSARY

ABDUAL is a LP abduction approach based on the dual program transformation, evaluated via tabled dual programs (Alferes et al., 2004a) under the Well-Founded Semantics (Gelder et al., 1991).

ACORDA is a LP system that implements Prospective Logic Programming (Pereira and Lopes, 2009), and features abduction (via even loops over negation), updating (based on EVOLP (Alferes et al., 2002a)), and preferences (based on Dell'Acqua and Pereira (2007)).

EPA is a subsequent development of ACORDA (Lopes, 2006; Pereira and Lopes, 2009), and shares its features, but its abduction mechanism is instead based on ABDUAL; the PROBABILISTIC EPA variant adds, on top of EPA, a LP probabilistic implementation of P-log (Baral et al., 2009).

EVOLP is a LP updating language, whose syntax extends that of generalized LP by allowing assertions of rules (either appearing in the head or in the body of a rule), and whose semantics is based on evolution stable models that capture an evolution of logic programs (Alferes et al., 2002a).

EVOLP/R is a LP updating technique, based on Dynamic Logic Programming (Alferes et al., 2000), but restricting updates to fluents only, and additionally employs incremental tabling (Saha, 2006; Swift, 2014) of fluents.

LP Logic Programming.

QUALM is a LP system that implements the joint tabling technique of contextual abduction and updating, plus counterfactual reasoning in LP.

TABDUAL is a LP abduction technique, based on ABDUAL (Alferes et al., 2004a), that employs tabling (Swift, 1999; Swift and Warren, 2012) in contextual abduction, in order to reuse priorly obtained (and tabled) abductive solutions, from one abductive context to another.

CHAPTER 1

INTRODUCTION

"Think about the Machines for a while, Stephen. They are robots, and they follow the First Law. But the Machines work not for any single human being, but for all humanity, so that the First Law becomes: 'No Machine may harm humanity; or, through inaction, allow humanity to come to harm."'
Isaac Asimov in *"The Evitable Conflict"*

1.1 Motivations

In recent years, systems or agents have become ever more sophisticated, autonomous, and act in groups, amidst populations of other agents, including humans. Autonomous robots or agents have been actively developed to be involved in a wide range of fields, where more complex issues concerning responsibility are in increased demand of proper consideration, in particular when the agents face situations involving choices on moral or ethical dimensions:[1]

- In medical or elder care, a robot may be confronted with conflicts between attaining its duties to treat the patient and respecting the patient's decision, e.g., in the case where the patient rejects a critical treatment the robot recommends.

- In the military, where robots from different makers, with diverse purposes, shapes and sizes have been built and even deployed in wars, will naturally face moral dilemmas, e.g., whether it is permissible for a drone to fire on a house where a target is known to be hiding, but at the same time it is one that also sheltering civilians. In fact, there has been much attention given recently concerning the ethical issue of such autonomous military robots (Arkin, 2009; Horowitz and Scharre, 2015; Shane, 2012;

[1] In this thesis, the words ethics/ethical and morality/moral will be used interchangeably.

Singer, 2009; The Economist, 2012a; Weiner, 2005). More recently, a multidisciplinary team of researchers received funding from the U.S. Navy to explore the challenges of providing autonomous agents with ethics (Higgins, 2014).

- Robotic prison guards, which have been under field trials, could also face conflicting situations in the handling of prisoners, e.g., concerning the involvement of harm in order to secure the prison.
- In a disaster recovery setting, where the robots involved in it face a tension in deciding who needs the most help most urgently, or between their duty of telling the truth and the risks of causing panic.
- Driverless cars, which may become more common in future, may also be subject to moral dilemmas: "Should a driverless car veer to prevent hitting crossing pedestrians even if that will cause hard collision with another car that will severely injure its occupants?", which sounds similar to the classic trolley problem dilemma introduced by Foot (1967).
- Moreover, teams of autonomous robots will need common ground for cooperating amongst themselves, even if originating from different makers.

As these demands become ever more pervasive and ubiquitous, the requirement that agents should function in an ethically responsible manner is becoming a pressing concern. Accordingly, *machine ethics*, which is also known under a variety of names, such as machine morality, computational morality, artificial morality, and computational ethics, emerges as a burgeoning field of inquiry to attend to that need, by imbuing autonomous agents with the capacity for moral decision making.

Moor (2011) proposes a hierarchical schema for categorizing ethical agents. At its lowest level are *ethical-impact agents*, which essentially refers to agents that can be evaluated by their ethical consequences. He picks the example of robotic camel jockeys that replace the role of enslaved young boys to ride the camels in camel racing (Lewis, 2005). At the next level are *implicit ethical agents*, which implicitly support ethical behavior, i.e., they are designed so that they do not pose negative ethical effects, typically by addressing issues like safety and critical reliability. For instance, an airplane's automatic pilot should warrant that the plane arrive safely. Then come *explicit ethical agents*, which are able to reason about ethics using ethical categories as part of their implementation. In Moor's viewpoint, this is the level towards which the emerging field of machine ethics is being driven at present. And finally, at the highest level are *full ethical agents*, which refer to those that can make and exercise explicit ethical judgments and generally are competent enough to reasonably justify them. This level – often associated with the requirement of agents to have some consciousness, intentionality, and free will – is the one that often broaches machine ethics into heated debate.

Clearly, machine ethics brings together perspectives from various fields, amongst them: philosophy, psychology, anthropology, evolutionary biology, and artificial intelligence. The

overall result of this interdisciplinary research is therefore not just important for equipping agents with some capacity for making moral decisions, but also to help better understand morality, via the creation and testing of computational models of ethical theories. The importance of working on machine ethics (in the sense of developing explicit ethical agents) is also emphasized in Moor (2011). First, ethics itself is important and machines that will ably treat us well are obviously desirable. Second, the increasing autonomy of agents with impacts on our life makes it indispensable to design agents with ethical principles governing their behavior. And third, programming ethics in a machine may help us to better understand ethics itself, as well as help create tutorial training programs.

Machine ethics as a field is increasingly more so recognized, and its importance has been emphasized in dedicated scientific meetings (e.g., Anderson et al. (2005a); Boissier et al. (2012); Calafate et al. (2015); Trappl (2013)), book publications (e.g., Anderson and Anderson (2011); Arkin (2009); Lin et al. (2012); Trappl (2015); Wallach and Allen (2009); White and Searle (2015)), as well as a heightened public awareness to its economic import (The Economist, 2012a; The Economist, 2012b). The Future of Life Institute (2015b) explicitly identifies machine ethics as one important research priorities in promoting Artificial Intelligence (AI) research, which is not only capable, but also robust and beneficial. More recently, this initiative, which is supported by top AI researchers from industry and academia, received a significant amount of funding to run a global research program aiming at those priorities (The Future of Life Institute, 2015a). The topic continues to receive wide attention in a variety of conferences, both soliciting papers and promoting panels.

Existing research in machine ethics covers a broad spectrum of investigation, as indicated by various approaches employed in the field as well as diverse moral theories or aspects being modeled. Case-based reasoning is employed in TRUTH-TELLER (McLaren and Ashley, 1995) and SIROCCO (McLaren, 2003), which implement a computational model for making ethical decision by comparing a problem to real or hypothetical cases, based on casuistry reasoning (Jonsen and Toulmin, 1988). In Anderson et al. (2005b), JEREMY is developed based upon the theory of act utilitarianism, to compute a morally right act by maximizing the pleasure of those affected by an act, by a simple summation of the components affecting the pleasure. Its subsequent development, W.D. (Anderson et al., 2005b), based on the theory of prima facie duties of Ross (1930), considers more duties to uphold, and computes a moral decision by a weighted summation, where a different weight can be assigned to each duty. The theory of prima facie duties is also considered in MEDETHEX (Anderson et al., 2006b) and ETHEL (Anderson and Anderson, 2008); both are implemented using Inductive Logic Programming (Muggleton, 1991) to learn from cases the morally right action based on the weight of the duties.

A more philosophical approach to machine ethics, without an implementation, is addressed in Powers (2006) by considering the first formulation of Kant's categorical imperative (Johnson, 2004). The author proposes to map universally quantified maxims onto categories in a deontic logic, and refers to non-monotonic logic to deal with the need

of commonsense reasoning in Kant's categorical imperative. In Bringsjord et al. (2006), mechanized deontic logic is envisaged as an approach for machine ethics, where ethically correct behaviors are shown by obtaining a formal proof of such behaviors within a sequent-based natural-deduction of an axiomatized utilitarian formulation of deontic logic (Murakami, 2004).

Another logic-based approach, using non-monotonic reasoning for addressing exceptions in a dilemma about lying is discussed in Ganascia (2007), where the answer set solver ANSPROLOG* (Baral, 2010) is used for representing and reasoning about the considered dilemma. Addressing exceptions in moral dilemmas is related to the moral particularism viewpoint, where moral decisions depend on cases; this is in contrast to moral generalism, which concerns a general application of moral principles to cases. The dispute between these two viewpoints is computationally studied in Guarini (2011) by using artificial neural networks, which are trained with cases about permissibility of actions involving killing and allowing to die.

Several logic-based approaches have been employed in the above machine ethics research, e.g., in Anderson and Anderson (2008); Anderson et al. (2006b); Bringsjord et al. (2006); Ganascia (2007); Powers (2006). While some approaches provide implementations in Logic Programming (LP) systems, such as in Anderson et al. (2006b), Ganascia (2007), and Anderson and Anderson (2008), they have not exploited LP-based reasoning features and recent techniques in LP systems that appear essential and promising for moral reasoning and decision making. Ganascia (2007) mainly just emphasizes the use of default negation in defeasible rules to capture non-monotonic reasoning, whereas the use of LP in Anderson et al. (2006b) and Anderson and Anderson (2008) is constrained to its purpose for learning rules from cases. Clearly, the potential of Logic Programming goes beyond that, and its appropriateness to machine ethics is begging to be explored.

1.2 Aims and Scope

This PhD thesis aims at investigating further the appropriateness of Logic Programming to machine ethics, notably by a combination of LP-based reasoning features, including techniques more recently made available in LP systems.

Given the broad and interdisciplinary nature of machine ethics, and the multitude of dimensions in ethics itself, this thesis is constrained to particular moral facets and viewpoints, which are nevertheless well-studied in philosophy and psychology. This choice of moral facets and viewpoints neither aims at defending them nor resolving their relevant moral dilemmas, as even philosophers may split opinions in their judgments. Instead, the purpose is to show that diverse LP-based reasoning features, in their combination, are capable and appropriate for expressing the considered viewpoints. To this end, the investigation does not merely address the appropriateness of LP-based reasoning abstractly, but also provides an implementation as a testing ground for experimentation of said moral facets. In other words, the investigation is not intended as a proposal for a machine readily

incorporating ethics, but as proof of concept that our understanding of the considered moral facets can in part be computationally modeled and implemented, testing them through a variety of classic moral examples taken off-the-shelf from the morality literature, with their reported empirical results about their judgments as a reference for validation.

1.2.1 Representing Morality

The first step of the investigation involves an interdisciplinary study from moral philosophy, psychology, and computer science. On the one hand, it requires reading philosophy and psychology literature to understand theories and results from this field. On the other hand, good familiarity with concepts from LP for knowledge representation and reasoning is expected. The result of this study is to identify which moral facets (and the viewpoints thereof) whose characteristics amenable to computational modeling, and which LP concepts are appropriate for representing these facets and reasoning about them.

There are three moral facets tackled in this thesis:

- Moral permissibility, taking into account viewpoints such as the Doctrines of Double Effect (McIntyre, 2004), Triple Effect (Kamm, 2006), and Scanlonian contractualism moral theory (Scanlon, 1998).

- The dual-process model (Cushman et al., 2010; Evans, 2010; Mallon and Nichols, 2010; Stanovich, 2011), which stresses the interaction between deliberative and reactive processes in moral decision making.

- Counterfactual thinking – thoughts on what could have happened, had some matter (action, outcome, etc.) been different in the past – and its role in moral reasoning, as studied, e.g., in Weiner (1995), Roese (1997), Byrne (2007), and Hoerl et al. (2011).

A thorough study of these moral facets allows us to identify a number of LP concepts that are appropriate for representing and reasoning about them, viz., abduction (with integrity constraints) (Alferes et al., 2004a; Kakas et al., 1992), preferences over abductive scenarios (Dell'Acqua and Pereira, 2007), probabilistic reasoning in LP (Baral et al., 2009; Han et al., 2008; Riguzzi and Swift, 2011), updating (Alferes et al., 2000; Alferes et al., 2002a; Alferes et al., 2005), and from tabling technique (Swift, 1999; Swift, 2014; Swift and Warren, 2012), as well as both Stable Model (Gelfond and Lifschitz, 1988) and Well-Founded Semantics (Gelder et al., 1991). Additionally, to apply counterfactuals for moral reasoning, we develop a novel LP approach for counterfactuals evaluation (based on Pearl (2009)) as part of this thesis contribution.

1.2.2 Engineering Morality

Apart from counterfactuals and the use of tabling technique in abduction and updating, other components have initially been featured in two existing systems developed earlier,

viz., ACORDA (Lopes, 2006; Lopes and Pereira, 2006; Pereira and Lopes, 2009) and PROBABILISTIC EPA (Han, 2009; Han et al., 2008; Pereira and Han, 2009). Both are implemented in XSB Prolog (Swift and Warren, 2012).

ACORDA is a system that implements Prospective Logic Programming (Pereira and Lopes, 2009), which enables an evolving program to look ahead prospectively into its possible future states and to prefer among them to satisfy its goals. It combines LP abduction, updating, and preferences over abductive scenarios. Its implementation is based on a meta-interpreter of evolving logic programs language EVOLP (Alferes et al., 2002a). An ad hoc abduction is implemented on top of it by means of even loops over negation. In XSB Prolog, which is based on the Well-Founded Semantics, such representation for abduction results in the so-called residual program (i.e., a program whose literals are undefined in its well-founded model), obtained subsequent to top-down query evaluation. Abductive stable models of this residual program are then computed by the *XSB Answer Set Programming (XASP)* package (Castro et al., 2015), which provides the candidates for a posteriori preferences. The implementation of a posteriori preferences reasoning in ACORDA is supported by the results from Dell'Acqua and Pereira (2007).

The Evolution Prospection Agent (EPA) system (Han, 2009; Pereira and Han, 2009) is an ulterior development of ACORDA, sharing its main features: abduction, updating, and preference. However, its abduction mechanism is based on the dual program transformation ABDUAL (Alferes et al., 2004a). EPA is implemented on top of the NEGABDUAL meta-interpreter; the latter is itself based on ABDUAL but with constructive negation feature (Ceruelo, 2009). Later, EPA is extended into PROBABILISTIC EPA by a probabilistic reasoning feature based on the probabilistic LP language P-log (Baral et al., 2009).

Taking into account the relevance of tabling and counterfactuals for representing considered moral facets, QUALM (its initial concept appears in Saptawijaya and Pereira (2014)) is developed, by focusing on two basic required features, viz., abduction and updating, now supported by tabling mechanisms. That is, it combines two LP engineering techniques developed in this thesis, viz., tabling abductive solutions in contextual abduction and incremental tabling of fluents for LP updating. QUALM also features counterfactual reasoning – which is not included in ACORDA nor in EPA– implementing our LP counterfactuals evaluation procedure.

The three systems (ACORDA, PROBABILISTIC EPA, and QUALM) are employed to model diverse issues of moral facets, depending on the need of their respective combination of features.

1.3 Thesis Main Contributions

There are two main contributions of this thesis.

1. Novel approaches for employing tabling in abduction and updating – individually

and combined – required to model various issues of moral facets identified in Section 1.2.1.

First, we conceptualize an approach for tabling abductive solutions in contextual abduction, called TABDUAL, for reusing priorly obtained (and tabled) abductive solutions, from one abductive context to another.

Second, incremental tabling (Saha, 2006; Swift, 2014), of XSB Prolog, is employed for LP updating in an approach called EVOLP/R, in order to correctly maintain the table of fluents due to changes made by incremental assertion of fluents. Both TABDUAL and EVOLP/R are based on and justified by the existing theoretical results of ABDUAL (Alferes et al., 2004a) and Dynamic Logic Programming (Alferes et al., 2000), respectively.

Third, we innovatively make use of LP abduction and updating in a procedure for evaluating counterfactuals, taking the established approach of Pearl (2009) as reference. The approach concentrates on pure non-probabilistic counterfactual reasoning in LP, by resorting to an interplay of abduction and updating, in order to determine the logical validity of counterfactuals.

Finally, a unified approach that seamlessly integrates TABDUAL and EVOLP/R is conceived, and implemented in a system prototype QUALM. Additionally, QUALM implements the above LP-based counterfactual evaluation procedure that requires the interplay between abduction and updating.

The new approaches introduced in this first contribution are of interest in themselves, and not specific to morality applications. They are more generally useful, and may be adoptable by systems other than XSB Prolog.

2. Applications of the three systems described in Section 1.2.2 to model various issues from the identified relevant moral facets.

   In the first application, ACORDA is utilized to model moral permissibility according to the Doctrines of Double Effect and Triple Effect through a number of cases from the classic trolley problem (Foot, 1967; Hauser et al., 2007; Hauser, 2007). In this application, integrity constraints in abduction are used for ruling out impermissible abduced actions *a priori*, whereas *a posteriori* preferences are employed to further prefer amongst the permissible actions those resulting in greater good. In this case, if an a priori integrity constraint corresponds to the agent's fast and immediate response, generating intended decisions that comply with deontological ethics (achieved by ruling out the use of intentional harm), then a posteriori preferences amongst permissible actions correspond to a slower response, as they involve more reasoning on action-generated models in order to capture utilitarianism (which favors welfare-maximizing behaviors), cf. the psychological empirical tests reported by Greene et al. (2004) and Cushman et al. (2010) in the dual-process model.

The second application adequately explores probabilistic moral reasoning, viz., to reason about actions, under uncertainty that might have occurred, and thence provide judgment adhering to moral principles within some prescribed uncertainty level. The case studied in this application, by means of PROBABILISTIC EPA, is jury trials in courts, which are required to proffer rulings beyond reasonable doubt with respect to moral permissibility according to the Doctrine of Double Effect. The commonly accepted probability of proof beyond reasonable doubt (e.g., see Newman (2006)) may serve as a common ground (in the sense of Scanlonian contractualism (Scanlon, 1998)) for guilty verdicts to be qualified as 'beyond reasonable doubt'. A form of argumentation may take place by presenting different evidences, via LP updating, as a consideration whether an exception can justify a different verdict.

In the third application, QUALM is employed to experiment with the issue of moral updating that allows other (possibly overriding) moral rules to be adopted by an agent, on top of those it currently follows. This application particularly demonstrates the direct use of LP updating so as to put an imposed moral rule into effect, and the conceptual benefit of tabling abductive solutions in contextual abduction to enact an interaction between deliberative and reactive processes occurring in the dual-process model.

Finally, the issue of moral permissibility with respect to the Doctrines of Double Effect and Triple Effect is revisited, now innovatively via counterfactuals, with QUALM as the vehicle of its experimentation. In this application, counterfactuals are particularly engaged to distinguish whether an effect of an action is a cause for achieving a morally dilemmatic goal or merely a side-effect of that action, such distinction is essential for establishing moral permissibility with respect to the Doctrine of Double Effect. The comparison to inspection points (Pereira et al., 2013) to alternatively express this distinction is illustrated. The application of counterfactuals is further extended to address moral justification: first, in the form of compound counterfactuals for justifying with hindsight a moral judgment that was passed under lack of current knowledge; and second, in the spirit of Scanlonian contractualism, by providing conceptual counterfactual queries for justifying exceptions to impermissibility of actions.

The above two main contributions are supported by our own relevant publications in conferences, journals, and invited book chapters. The citations below refer to the publications listed at the end of this chapter. Note that some specific contributions above result from the refinement of their respective publications.

- The first main contribution is supported by the following publications:
  Pereira and Saptawijaya (2012); Pereira and Saptawijaya (2015a); Saptawijaya and Pereira (2013b); Saptawijaya and Pereira (2013c); Saptawijaya and Pereira (2013d);

Saptawijaya and Pereira (2013e); Saptawijaya and Pereira (2013f); Saptawijaya and Pereira (2013g); Saptawijaya and Pereira (2014a); Saptawijaya and Pereira (2015b).

- The second main contribution is supported by the following publications:
  Han et al. (2012); Pereira and Saptawijaya (2007a); Pereira and Saptawijaya (2007b); Pereira and Saptawijaya (2009a); Pereira and Saptawijaya (2009b); Pereira and Saptawijaya (2011); Pereira and Saptawijaya (2015a); Pereira and Saptawijaya (2015c); Pereira and Saptawijaya (2015d); Pereira and Saptawijaya (2015e); Saptawijaya and Pereira (2015a); Saptawijaya and Pereira (2015c).

- Other publications that support chapters in this thesis:
  Pereira and Saptawijaya (2015b); Pereira and Saptawijaya (2015f); Saptawijaya (2013); Saptawijaya and Pereira (2013a); Saptawijaya and Pereira (2014b); Saptawijaya and Pereira (2015c).

1.4 Structure of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 presents a survey of research in machine ethics, which provides the context and the motivation for employing LP-based representation and reasoning in machine ethics.

- Chapter 3 reports on our literature study in moral philosophy and psychology for choosing moral facets and their conceptual viewpoints that are close to LP-based representation and reasoning.

- Chapter 4 provides necessary background of Logic Programming and subsequently discusses the appropriateness of various LP concepts for representing and reasoning about diverse issues of moral facets tackled in this thesis.

- Chapter 5 details novel approaches for employing tabling in abduction and updating, viz., tabling abductive solutions in contextual abduction (TABDUAL) and the incremental tabling of fluents (EVOLP/R), respectively.

- Chapter 6 elaborates our LP-based counterfactuals evaluation procedure, concentrating on pure non-probabilistic counterfactual reasoning by resorting to abduction and updating, in order to determine the logical validity of counterfactuals.

- Chapter 7 discusses the three LP systems (ACORDA, PROBABILISTIC EPA, and QUALM), emphasizing on how each of them distinctively incorporates a combination of LP-based representation and reasoning features discussed in Chapter 4.

- Chapter 8 details the applications of ACORDA, PROBABILISTIC EPA, and QUALM in modeling various issues relevant to the chosen moral facets.

- Chapter 9 concludes the thesis and lists potential future work.

1.5 Reading Paths

The thesis is best read sequentially, chapter by chapter. Nevertheless, several alternative reading paths are possible, as shown below. Those in parentheses provide some necessary background, technical details, and further references for the respective topic. They may safely be skipped and read only if needed.

- A general survey of approaches to machine ethics:

$$2 \rightarrow 3 \rightarrow 4 \rightarrow 9$$

- General LP engineering techniques:
  - Abduction with tabling: $4.1 \rightarrow (4.2) \rightarrow (4.7) \rightarrow 5.1$
  - Updating with tabling: $4.1 \rightarrow (4.5) \rightarrow (4.7) \rightarrow 5.2$
  - Counterfactuals: $4.1 \rightarrow (4.6) \rightarrow 6$

- Modeling morality with focus on:
  - Abduction and preferences over abductive scenarios:

$$(3) \rightarrow 4.1 \rightarrow 4.2 \rightarrow 4.3 \rightarrow 7.1 \rightarrow 8.1$$

  - Abduction and probabilistic LP:

$$(3) \rightarrow 4.1 \rightarrow 4.2 \rightarrow 4.4 \rightarrow 7.2 \rightarrow 8.2$$

  - Abduction, updating, and tabling:

$$(3) \rightarrow 4.1 \rightarrow 4.2 \rightarrow 4.5 \rightarrow 4.7 \rightarrow 5 \rightarrow 7.3 \rightarrow 8.3$$

  - Counterfactuals:

$$(3) \rightarrow 4.1 \rightarrow 4.2 \rightarrow 4.5 \rightarrow 4.6 \rightarrow 6 \rightarrow (7.3) \rightarrow 8.3.2$$

List of Publications

Han, T. A., A. Saptawijaya, and L. M. Pereira (2012). "Moral Reasoning Under Uncertainty". In: *Procs. 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*. Vol. 7180. LNCS. Springer, pp. 212–227.

Pereira, L. M. and A. Saptawijaya (2007a). "Modelling Morality with Prospective Logic". In: *Procs. 13th Portuguese International Conference on Artificial Intelligence (EPIA)*. Vol. 4874. LNAI, pp. 99–111.

Pereira, L. M. and A. Saptawijaya (2007b). "Moral Decision Making with ACORDA". In: *Local Procs. 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*.

Pereira, L. M. and A. Saptawijaya (2009a). "Computational Modelling of Morality". In: *The Association for Logic Programming Newsletter* 22(1).

Pereira, L. M. and A. Saptawijaya (2009b). "Modelling Morality with Prospective Logic". In: *International Journal of Reasoning-based Intelligent Systems* 1(3/4), pp. 209–221.

Pereira, L. M. and A. Saptawijaya (2011). "Modelling Morality with Prospective Logic". In: *Machine Ethics*. Ed. by M. Anderson and S. L. Anderson. Cambridge University Press: New York, NY, pp. 398–421.

Pereira, L. M. and A. Saptawijaya (2012). "Abductive Logic Programming with Tabled Abduction". In: *Procs. 7th International Conference on Software Engineering Advances (ICSEA)*. ThinkMind, pp. 548–556.

Pereira, L. M. and A. Saptawijaya (2015a). *Abduction and Beyond in Logic Programming with Application to Morality*. Accepted at *Frontiers of Abduction, a special issue of IfCoLog Journal of Logics and their Applications*. Available from (preprint): `http://goo.gl/yhmZzy`.

Pereira, L. M. and A. Saptawijaya (2015b). "Bridging Two Realms of Machine Ethics". In: *Rethinking Machine Ethics in the Age of Ubiquitous Technology*. Ed. by J. B. White and R. Searle. IGI Global.

Pereira, L. M. and A. Saptawijaya (2015c). "Counterfactuals in Critical Thinking, with Application to Morality". In: *Procs. 2nd International Seminar on Critical Thinking (SIPC)*. Available from: `http://http://goo.gl/Og5iA3`.

Pereira, L. M. and A. Saptawijaya (2015d). "Counterfactuals in Critical Thinking, with Application to Morality (extended abstract)". In: *Procs. International Conference on Model-Based Reasoning in Science and Technology (MBR)*. Available from: `http://goo.gl/CbxbvO`.

Pereira, L. M. and A. Saptawijaya (2015e). *Counterfactuals in Logic Programming with Applications to Agent Morality*. Accepted at a special volume of *Logic, Argumentation & Reasoning*. Available from (preprint): `http://goo.gl/6ERgGG`.

Pereira, L. M. and A. Saptawijaya (2015f). "Software with Ethical Discernment (extended abstract)". In: *Procs. International Conference on Model-Based Reasoning in Science and Technology (MBR)*. Available from: `http://goo.gl/9LaldX`.

Saptawijaya, A. (2013). "Towards Computational Morality with Logic Programming (Technical Communication in 29th International Conference on Logic Programming (ICLP) Doctoral Consortium)". In: *Theory and Practice of Logic Programming, Online Supplement* 13(4-5). Available from: `http://journals.cambridge.org/downloadsup.php?file=/tlp2013037.pdf`.

Saptawijaya, A. and L. M. Pereira (2013a). "Exploiting Logic Programming as a Computational Tool to Model Morality". In: *Austrian Research Institute for Artificial Intelligence OFAI's Workshop on "A Construction Manual for Robots' Ethical Systems: Requirements, Methods, Implementation, Tests"*. Available from: `http://goo.gl/e2M0ZT`.

Saptawijaya, A. and L. M. Pereira (2013b). "Implementing Tabled Abduction in Logic Programs". In: *Local Procs. 16th Portuguese International Conference on Artificial Intelligence (EPIA)*. Doctoral Symposium on Artificial Intelligence (SDIA), pp. 518–529.

Saptawijaya, A. and L. M. Pereira (2013c). "Incremental Tabling for Query-Driven Propagation of Logic Program Updates". In: *Procs. 19th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*. Vol. 8312. LNCS. Springer, pp. 694–709.

Saptawijaya, A. and L. M. Pereira (2013d). "Program Updating by Incremental and Answer Subsumption Tabling". In: *Procs. 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Vol. 8148. LNCS. Springer, pp. 479–484.

Saptawijaya, A. and L. M. Pereira (2013e). "Tabled Abduction in Logic Programs (Technical Communication in 29th International Conference on Logic Programming (ICLP))". In: *Theory and Practice of Logic Programming, Online Supplement* 13(4-5). Available from: `http://journals.cambridge.org/downloadsup.php?file=/tlp2013008.pdf`.

Saptawijaya, A. and L. M. Pereira (2013f). "Towards Practical Tabled Abduction in Logic Programs". In: *Procs. 16th Portuguese Conference on Artificial Intelligence (EPIA)*. Vol. 8154. LNAI. Springer, pp. 223–234.

Saptawijaya, A. and L. M. Pereira (2013g). "Towards Practical Tabled Abduction Usable in Decision Making". In: *Procs. 5th KES International Conference on Intelligent Decision Technologies (IDT)*. Frontiers of Artificial Intelligence and Applications (FAIA). IOS Press, pp. 429–438.

Saptawijaya, A. and L. M. Pereira (2014a). "Joint Tabling of Logic Program Abductions and Updates (Technical Communication in 30th International Conference on Logic Programming (ICLP))". In: *Theory and Practice of Logic Programming, Online Supplement* 14(4-5). Available from: `http://arxiv.org/abs/1405.2058`.

Saptawijaya, A. and L. M. Pereira (2014b). "Towards Modeling Morality Computationally with Logic Programming". In: *Procs. 16th International Symposium on Practical Aspects of Declarative Languages (PADL)*. Vol. 8324. LNCS. Springer, pp. 104–119.

Saptawijaya, A. and L. M. Pereira (2015a). "Logic Programming Applied to Machine Ethics". In: *Procs. 17th Portuguese International Conference on Artificial Intelligence (EPIA)*. Vol. 9273. LNAI. Springer.

Saptawijaya, A. and L. M. Pereira (2015b). "TABDUAL: a Tabled Abduction System for Logic Programs". In: *IfCoLog Journal of Logics and their Applications* 2(1), pp. 69–123.

Saptawijaya, A. and L. M. Pereira (2015c). "The Potential of Logic Programming as a Computational Tool to Model Morality". In: *A Construction Manual for Robots' Ethical Systems: Requirements, Methods, Implementations*. Ed. by R. Trappl. Cognitive Technologies. Springer.

CHAPTER 2

RESEARCH IN MACHINE ETHICS

In this chapter, a survey of research in machine ethics is presented, providing the context and the motivation for employing LP-based representation and reasoning in this field. Taking into account the relevance of problem domain investigated in this thesis, the survey is focused on the individual realm of machine ethics, where computation is a vehicle for representing moral cognition of an agent and its reasoning thereof.

The reader is referred to Pereira and Saptawijaya (2015) for a complementary survey of research in the collective realm of machine ethics, which emphasizes instead the emergence, in a population, of evolutionarily stable moral norms, of fair and just cooperation, that ably discards free riders and deceivers, to the advantage of the whole evolved population.

For a supplementary general philosophical viewpoint, as well as a survey in the same collective setting, on the issue of software sans "emotions" but with ethical discernment, the reader is referred to Pereira (2015).

2.1 TRUTH-TELLER and SIROCCO

TRUTH-TELLER (McLaren and Ashley, 1995) is a system that qualitatively compares a pair of ethical dilemma cases about whether to tell the truth and extracts ethically salient similarities and differences of the reasons for telling the truth (or not), from the perspective of the agent faced with the dilemma. The representation of a case is manually constructed from the interpretation of the story. Semantic networks are employed to represent the truth telling episodes (including the actors involved, their relationships, possible actions and reasons supporting possible actions), a hierarchy of relationships (familial, commercial, etc.), and a hierarchy of reasons for or against telling the truth based on the formulation in Bok (1989). The representation is then analyzed by case-based (casuistic) reasoning in several steps:

- First, a mapping between the reasons in two cases is built, viz., by matching similar and marking distinct reasons.
- The second step qualifies: (1) the relationships among actors, actions, and reasons in one case; and (2) the mappings of these objects to those in the other considered case based on considerations such as criticalness, actors' roles and alternative actions.
- The third step points out similarities and differences of cases, with respect to predefined comparison contexts, whether the considered reasons apply to both cases, apply more strongly in one case than another, or apply to only one case.

The analysis result is then summarized in a comparison text.

SIROCCO (McLaren, 2003) also employs case-based (casuistic) reasoning, but unlike TRUTH-TELLER, it accepts an ethical dilemma case and retrieves ethics principles and past cases that are relevant to the target case. It is developed in order to operationalize general abstract ethics principles, as ethicists often record their explanations of how and why they applied and reconciled principles in resolving specific cases. SIROCCO particularly addresses a domain of engineering ethics, taking into account ethics code of National Society of Professional Engineers (NSPE) (1996) and the cases decided by its Board of Ethical Review (BER). The BER's decision making indicates that several operationalization techniques are applied, which include linking ethics code and past cases with the facts of the considered case, grouping codes and cases so they can be cited in support of a conclusion, and reusing some reasoning applied in past cases to the context of a new case. In SIROCCO, cases are represented in general using the Ethics Transcription Language ETL (McLaren, 2003) as chronological narratives of facts involving actors, actions participated by actors, and temporal relations between facts. The representation of a source case is particularly extended with its analysis by BER, which captures an operationalization of NSPE ethics codes. In its retrieval process, SIROCCO first computes the best N matches of source cases with respect to the ETL fact matching between the target case and each source case, and subsequently finds a structural mapping using A^* search between the target case and the best N matches (Branting, 2000). The goal is to map facts of a source to a corresponding fact in the target at the same level of abstraction, while keeping a consistent mapping between their actors and temporal relations. Finally, it analyzes the results of multiple source cases (rather than of a single best match) to generate suggestions for the target case, such as relevant principles and relevant source cases.

2.2 JEREMY and W.D.

JEREMY (Anderson et al., 2005b) is a system that follows the theory of act utilitarianism. This theory maintains that an act is morally right if and only if the act maximizes the good, viz., the one with the greatest net good consequences, taking into account all those affected by the action (Sinnott-Armstrong, 2003). In JEREMY, hedonistic act utilitarianism

is particularly adopted, where the pleasure and displeasure of those affected by each possible action are considered. This is manifested by three components with respect to each affected person p: (1) the intensity I_p of pleasure/displeasure, scaled between -2 to 2; (2) the duration D_p of the pleasure/displeasure, in days; and (3) the probability P_p that this pleasure/displeasure will occur. The total net pleasure for each action a is computed as follows:

$$Total_a = \Sigma_{p \in Person}(I_p \times D_p \times P_p).$$

The right action is the one giving the highest total net pleasure $Total_a$.

In order to respond to critics of act utilitarianism, another prototype, W.D. (Anderson et al., 2005b), is developed to avoid a single absolute duty. That is, it follows several duties, where in different cases a duty can be stronger than (and thus overrides) the others, following the theory of prima facie duties of Ross (1930). This theory comprises duties like fidelity (one should honor promises), reparation (one should make amends for wrongs done), gratitude (one should return favors), justice (one should treat people as they deserve to be treated), beneficence (one should act so as to bring about the greatest good), non-maleficence (one should act so as to cause the least harm), and self-improvement (one should develop one's own abilities/talent to the fullest).

In W.D., the strength of each duty is measured by assigning it a weight, capturing the view that a duty may take precedence over another. W.D. computes, for each possible action, the weighted sum of duty satisfaction, and returns the greatest sum as the right action. In order to improve the decision, in the sense of conforming to a consensus of correct ethical behavior, the weight of a duty is allowed to be adjusted through a supervised learning, by acquiring suggested action from the user. This weight adjustment to refine moral decision is inspired by reflective equilibrium of Rawls (1971): reflecting on considered judgments about particular cases and revising any elements of these judgments (principles that govern these judgments, theories that bear on them, etc.) wherever necessary, in order to achieve an acceptable coherence amongst them, the so-called equilibrium (Daniels, 2003). In Anderson et al. (2005b), it is however unclear which supervised learning mechanism is actually implemented in W.D.

2.3 MEDETHEX and ETHEL

The theory of prima facie duties is further considered in Anderson et al. (2006b) and Anderson and Anderson (2008), while also concretely employing machine learning to refine its decision making. As in W.D., the employing of machine learning is also inspired by reflective equilibrium of Rawls (1971), viz., to generalize intuition about particular cases, testing this generalization on further cases, and then repeats this process to further refine the generalization towards the end of developing a decision procedure that agrees with intuition.

The first implementation is MEDETHEX (Anderson et al., 2006b), which is based on a more specific theory of prima facie duties, viz., the Principle of Biomedical Ethics of

Beauchamp and Childress (1979). The considered cases are a variety of the following type of ethical dilemma (Anderson et al., 2006a):

A healthcare professional has recommended a particular treatment for her competent adult patient, but the patient has rejected it. Should the healthcare professional try to change the patient's mind or accept the patient's decision as final?

The cases thus involve only two possible actions, viz., (1) accepting a patient's decision to reject a treatment; and (2) trying to convince him to change his mind. Furthermore, the cases are constrained to three of four duties in Beauchamp and Childress (1979), viz., respecting for the autonomy of the patient, not causing harm to the patient (non-maleficence), and promoting patient welfare (beneficence).

MEDETHEX is implemented using Inductive Logic Programming (ILP) (Muggleton, 1991) to learn the relation $supersedes(A_1, A_2)$, i.e., whether action A_1 supersedes (viz., is ethically preferable to) action A_2. The training (positive) examples comprise cases, where each case is associated with an estimate satisfaction/violation value of each duty for each possible action (scaled from -2 to 2) and the ethically preferred action for the case. The negative examples are obtained by simply exchanging the preferred action from the positive training examples. The relation $supersedes(A_1, A_2)$ is then learned from these positive and negative examples, expressing it in terms of the lower bounds for difference of values of the considered duties between the two actions A_1 and A_2.

Similar to MEDETHEX, ETHEL is also based on Beauchamp and Childress (1979), but applied to the domain of eldercare with the main purpose to remind a patient to take his/her medication, taking ethical duties into consideration. It also decides, after a patient has been reminded, whether to accept his/her refusal to take the medication (in which case a further reminder may take place) or to notify an overseer (e.g., a medical doctor) instead.

ETHEL is also implemented using ILP, following a similar technique employed in MEDETHEX to learn the same relation $supersedes(A_1, A_2)$; this relation is also defined in terms of the lower bounds for difference of values of the corresponding duties between actions A_1 and A_2. Unlike MEDETHEX, due to the reminder feature, the satisfaction/violation values of duties for each action in ETHEL are adjusted over time. This adjustment is determined by several factors, such as the maximum amount of harm if the medication is not taken, the number of hours for this maximum harm to occur, etc.; this information is obtained from the overseer. Adjusting the satisfaction/violation values of duties permits ETHEL to remind (or not) the patient to take his/her medication as well as to notify (or not) the overseer at ethically justifiable moment.

ETHEL has been deployed in a robot prototype, 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 (Anderson and Anderson, 2010).

2.4 A Kantian Machine Proposal

A more philosophical tone of machine ethics is presented in Powers (2006), where he argues that rule-based ethical theories like the first formulation of Kant's categorical imperative ("Act only according to that maxim whereby you can at the same time will that it should become a universal law without contradiction" (Kant, 1981)) appear to be promising for computational morality, because of their computational structure for judgment. Three views on how to computationally model categorical imperative are envisaged.

First, in order for a machine to maintain consistency in testing ethical behavior, it should be able to construct a moral theory that renders individual maxims to be universally quantified (over circumstances, purposes, and agents) and to map them onto deontic categories, viz., forbidden, permissible, and obligatory action. Deontic logic is regarded as an appropriate formalism with respect to this first view. He abstractly refers to schemata for the three deontic categories, that for every agent, circumstance C, and purpose P:

- Action A is obligatory: $(C \text{ and } P) \rightarrow A$.
- Action A is forbidden: $(C \text{ and } P) \rightarrow \neg A$.
- Action A is permissible: $\neg((C \text{ and } P) \rightarrow A)$ and $\neg((C \text{ and } P) \rightarrow \neg A)$.

where a candidate maxim should be an instance of these three schemata.

Powers suggests that mere consistency is not sufficient for a maxim. Instead, its consistency should also be checked with other existing facts or background theory. This leads to his second view, viz., the need of commonsense reasoning in the categorical imperative to deal with contradiction. For this view, he refers to non-monotonic logic, which is appropriate to capture defeating conditions to a maxim. In this regard, he particularly resorts to default logic of Reiter (1980) as a suitable formalism, that adding the default rules allows maxims to contradict the background set of facts and commonsense rules without introducing inconsistency.

In his third view, Powers contemplates on the construction of a coherent system of maxims, where he sees such construction analogous to the belief revision problems. In the context of bottom-up construction, he envisages an update procedure for a machine to update its system of maxims with another maxim, though it is unclear to him how such an update can be accomplished.

The formalisms in these three views are only considered abstractly and no implementation is referred to address them.

2.5 Machine Ethics via Theorem Proving

In Bringsjord et al. (2006), mechanized multi-agent deontic logic is employed with the view that ethically correct robot behaviors are those that can be proved in a deontic logic.

For obtaining such a proof of ethically permissible actions, they resort to a sequent-based natural-deduction of Murakami (2004) axiomatization of Horty's utilitarian formulation of multi-agent deontic logic (Horty, 2001). This deduction system is encoded in the interactive theorem prover ATHENA (Arkoudas et al., 2005). The use of *interactive* theorem prover is motivated by the idea that an agent operates according to ethical codes bestowed on them, and when its automated reasoning fails, it suspends its operation and asks human guidance to resolve the issue.

Taking an example in health care, where two agents are in charge of two patients with different needs (patient H_1 depends on life support, whereas patient H_2 on very costly pain medication), two actions are considered: (1) terminate H_1's life support to secure his organ for five humans; and (2) delay delivery of medication to H_2 to conserve hospital resources. The approach in Bringsjord et al. (2006) begins with supposing several candidates of ethical codes, from harsh utilitarian (that both terminates H_1's life and delay H_2 medication) to most benevolent (neither terminates H_1's life nor delay H_2 medication); these ethical codes are formalized using the aforementioned deontic logic. The logic additionally formalizes behaviors of agents and their respective moral outcomes. Given these formalizations, ATHENA is employed to query each ethical code candidate in order to decide which amongst them should be operative, meaning that the best moral outcome (viz., that resulting from neither terminates H_1's life nor delay H_2 medication) is provable from the operative one.

2.6 Particularism vs. Generalism

A computational model to study the dispute between particularism and generalism, is explored in Ganascia (2007) and Guarini (2011). Moral generalism stresses the importance of moral principles and their *general* application in moral reasoning, whereas moral particularism favors on the view that moral reasoning (and decisions) depend on cases and not on a general application of moral principles to cases (Dancy, 2001).

In Ganascia (2007), different ethical principles (of Aristotle, Kant, and Benjamin Constant) are modeled using answer set programming, implemented with ANSPROLOG* (Baral, 2010). The aim is to show that non-monotonic logic is appropriate to address the opposition between generalism and particularism by capturing justified exceptions in general ethics rules. The tension between these two viewpoints is exemplified by a classic dilemma about lying: in a war situation one hides a friend who is wanted by the military force, raising to a dilemma whether he should tell the truth, denouncing his friend to the military, which leads to the murder of his friend.

In order to model this dilemma in the view of Aristotle's ethics (viz., choosing the least unjust action), several possible actions are conceived, e.g., tell the truth, tell a lie, etc., and facts about consequences of these actions are defined. Predicate $unjust(A)$ is then defined by assessing whether the consequence of A is worse than the consequence of other actions, via predicate $worse/2$, whose parameters are the consequences of two considered

actions. Depending on the definition of $worse/2$, the answer sets may be split into one part corresponding to telling the truth and the other part to telling a lie. The model itself does not provide a mechanism to prefer among these answer sets, though it illustrates that an ad hoc preference is possible by explicitly changing the definition of $worse/2$ predicate so as all answer sets contain the action of telling a lie (providing that murder has worse consequence than that of all other actions).

Ganascia (2007) also contrasts Kant's categorical imperative and Constant's objection. For Kant's categorical imperative, a rule such as:

$$act(P, A) \leftarrow person(P), action(A), act(\text{“I”}, A)$$

is defined to universalize a maxim: it stipulates that if "I" act in such A, all person(P) could act the same. This view does not require preferences among different actions, but emphasizes possible consequences of a maxim that cannot be universalized, e.g., a society where nobody can be trusted: $untrust(P) \leftarrow act(P, tell(P, lie))$. To this end, while lying can be admitted in an answer set, the answer set reflects a world where nobody can be trusted.

While this Kantian view aims at upholding generality of ethics principles, Constant's theory authorizes principles that tolerate exceptions. The lying dilemma is modeled by capturing a more specific principle for telling the truth: we always have to tell the truth, except to someone who does not deserve it. This is achieved in the model by: (1) not only considering the transmitter of the speech (as in the Kantian model), but also the receiver; and (2) using default negation to express the principle in a way that one should always tell the truth, except when the receiver is a murderer.

In Guarini (2011), the dispute between particularism and generalism is addressed using artificial neural networks. More specifically, simple recurrent networks are trained with cases about permissibility of actions involving killing and allowing to die.

The input for a network encodes the actor, the recipient of the action and the motive or the consequence of the action (e.g., killing in self-defence, allowing one to die to save many innocents, etc.), but without the provision of explicit moral rules, whereas the output of the network determines the permissibility of an input case. The experiments are performed on several networks that are trained differently. For instance, one network is trained by classifying permissibility based on the motive or the consequence of the action (irrespective whether the action is killing or allowing to die), whereas another network is trained by distinguishing the action killing from allowing to die.

By employing these trained networks to classify test cases, one result suggests that acting in self-defence contributes to permissibility, whereas actions that lead to the deaths of innocents are impermissible. Further analysis on the similarity of hidden unit activation vector between cases suggests that killing and allowing to die are making different contributions to the similarity spaces for different trained networks. Nonetheless, the networks admittedly learn some principles in general, though it cannot be directly expressed in classical, discrete representational structure. The experiments thus show that the behavior

of the networks is in agreement with the so-called *contributory standards* of moral principles (Dancy, 2001) – a middle ground between particularist and generalist – which allows more than one principle to be applicable to a case, as each specifies how things are, only in a certain respect.

2.7 Concluding Remarks

In this chapter the open and broad nature of research in machine ethics has been illustrated. On the one hand, it spans a variety of morality viewpoints, e.g., utilitarianism, Kant's categorical imperative, prima facie duties, particularism and generalism. On the other hand, a number of approaches have been proposed to model these viewpoints, with some assumptions to simplify the problem, which is somewhat unavoidable, given the complexity of human moral reasoning. The open nature of this research field is also indicated by different purposes these approaches are designed for (e.g., retrieving similar moral cases, explicitly making moral decisions, or finding operative moral principles).

TRUTH-TELLER and SIROCCO point out the role of knowledge representation (using semantic networks and its specific language ETL, respectively) to represent moral cases in a sufficiently fine level of detail, and rely on such representation for comparing cases and retrieving other similar cases. This form of representation is not so emphasized in JEREMY and W.D., as they reduce the utilitarian principle and duties into numerical values within some scale. Unlike TRUTH-TELLER and SIROCCO, JEREMY and W.D. aim at explicitly making moral decisions; these decisions are determined by these values through some procedure capturing the moral principles followed. Such quantitative valuation of duties also forms the basis of MEDETHEX and ETHEL, though some basic LP representation is employed for representing this valuation in positive and negative instances (which are needed for their ILP learning mechanism), as well as for representing the learned principle in the form of a LP rule. This learned principle, in terms of these numerical values, determines moral decisions made by these systems.

The employment of logic-based formalisms in the field, notably deontic logic, to formalize moral theory appears in the Kantian machine proposal of Powers (2006). Indeed, the insufficient of abstract logic-based formalism for rule-based ethical theories is identified in this proposal, emphasizing the need of non-monotonic reasoning in order to capture defeating conditions to a maxim. Moreover, the proposal also points out the importance of an update procedure to anticipate updating a system of maxims with another. Unfortunately there is no concrete realization of this proposal. In Bringsjord et al. (2006), an interactive theorem prover is employed to encode a specific deontic logic formalism in order to find an operative moral principle (amongst other available ones) in the form of proof. The use of theorem prover in this approach however does not concern the non-monotonic reasoning and the moral updating issues raised by Powers (2006).

The issue of non-monotonic reasoning becomes more apparent in the study about particularism vs. generalism. Ganascia (2007) demonstrates in a concrete moral case

how non-monotonic reasoning can be addressed in LP – more specifically in answer set programming – using defeasible rules and default negation to express principles that tolerate exception. From a different perspective, the experiments with artificial neural networks (Guarini, 2011) also reveal that more than one principle may be applicable to similar cases that differ in a certain aspect (e.g., motives, consequences, etc.), thus upholding morality viewpoints that tolerate exceptions.

While the survey in this chapter shows that several logic-based approaches have been employed in machine ethics, the use of LP has not been much explored in the field despite its potential:

- Like TRUTH-TELLER and SIROCCO, LP permits declarative knowledge representation of moral cases with sufficiently level of detail to distinguish one case from other similar cases. Indeed, except the philosophical approach by Powers (2006), all other approaches anchor the experiments to concrete moral cases, indicating that representing moral principles alone is not enough, but the principles need to be materialized into concrete examples. Clearly, the expressivity of LP may extend beyond basic representation of (positive/negative) example facts demonstrated in MEDETHEX and ETHEL.

- Given its declarative representation of moral cases, appropriate LP-based reasoning features can be employed for moral decision making, without being constrained merely to quantitative simplifying assumption (cf. MEDETHEX and ETHEL) and ILP. For instance, the role of LP abduction (Kakas et al., 1998) for decision making in general is discussed in Kowalski (2011). Indeed, LP abduction has been applied in a variety of areas, such as in diagnosis (Gartner et al., 2000), planning (Eshghi, 1988), scheduling (Kakas and Michael, 2001), reasoning of rational agents and decision making (Kowalski and Sadri, 2011; Pereira et al., 2013), knowledge assimilation (Kakas and Mancarella, 1990), natural language understanding (Balsa et al., 1995), security protocols verification (Alberti et al., 2005), and systems biology (Ray et al., 2006). These applications demonstrate the potential of abduction, and it may as well be suitable for moral decision making, albeit without focusing on learning moral principles. Moral reasoning with quantitative valuation of its elements (such as actions, duties, etc.), either in utility or probability, can still be achieved with other LP-based reasoning features in combination with abduction, e.g., using preferences (see, e.g., Dell'Acqua and Pereira (2007)) and probabilistic LP (see, e.g., Baral et al. (2009); Riguzzi and Swift (2011)).

- LP provides a logic-based programming paradigm with a number of practical Prolog systems, allowing not only addressing morality issues in an abstract logical formalism (e.g., deontic logic in Powers (2006)), but also via a Prolog implementation as proof of concept and a testing ground for experimentation. The use of a theorem prover in Bringsjord et al. (2006) to find a proof of an operative moral principle with

respect to a particular deontic logic is an attempt to provide such a testing ground for experimentation, albeit not addressing non-monotonic reasoning and moral updating concerns of Powers (2006). The use of LP, without resorting to deontic logic, to model Kant's categorical imperative and non-monotonic reasoning (via default negation), is shown in Ganascia (2007), but no LP updating is considered yet. To this end, a combination of LP abduction and updating may be promising in order to address moral decision making with non-monotonic reasoning and moral updating, in line with the views of Powers (2006).

- While logical formalisms, such as deontic logic, permit to specify the notions of obligation, prohibition and permissibility in classic deontic operators, they are not immediately appropriate for representing morality reasoning *processes* studied in philosophy and cognitive science, such as the dual-process model of reactive and deliberative processes (Cushman et al., 2010; Evans, 2010; Mallon and Nichols, 2010; Stanovich, 2011). Advanced techniques in Prolog systems, such as tabling (Swift, 1999; Swift, 2014; Swift and Warren, 2012), open an opportunity to conceptually capture such processes, by appropriately applying it to considered reasoning mechanisms in moral decision making, such as LP abduction and updating.

Given this potential of LP in addressing all the above issues, there is a need to investigate further its potential. But before we do so, we need first to study more results from morality-related fields, such as philosophy and psychology, to better identify some significant moral facets which, at the start, are amenable to computational modeling by LP knowledge representation and reasoning features. This is the subject of the next chapter.

CHAPTER 3

SIGNIFICANT MORAL FACETS AMENABLE TO LOGIC PROGRAMMING

This chapter reports on our literature study in moral philosophy and psychology for choosing conceptual viewpoints close to LP-based reasoning. These viewpoints fall into three moral facets tackled in this thesis. In Section 3.1 we study *moral permissibility*, taking into account the Doctrines of Double Effect (McIntyre, 2004), Triple Effect (Kamm, 2006), and Scanlonian contractualism (Scanlon, 1998). We look into *the dual-process model* (Cushman et al., 2010; Evans, 2010; Mallon and Nichols, 2010; Stanovich, 2011), in Section 3.2, that stresses the interaction between deliberative and reactive processes in delivering moral decisions. Finally, in Section 3.3 we discuss the role of *counterfactual* thinking in moral reasoning.

3.1 Moral Permissibility

A trolley, whose conductor has fainted, is headed toward five people walking on the track. The banks of the track are so steep that these five people will not be able to get off the track in time. 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?

This is a well-known moral problem, called *the trolley problem*, originally introduced in Foot (1967).[1] It concerns itself the question of moral permissibility in a dilemma involving harm. While most people tend to agree, as shown in psychological empirical tests (e.g., in Hauser et al. (2007)), that it is permissible for Hank to throw the switch for saving the five

[1]The descriptions of various trolley problem cases in this section (and Section 8.1) are taken from Mikhail (2007).

(albeit killing one), it is not trivial for people to explain the moral rules that justify their moral judgment of this permissibility.

3.1.1 The Doctrines of Double Effect and Triple Effect

It is appealing for such dilemma to explain the permissibility of Hank's judgment by resorting to utilitarianism, which favors an action that maximizes utility as being moral. It explains in this trolley problem that the action of Hank to throw the switch is permissible, as it will kill less people compared to the other consequence of letting the runaway trolley to hit the five. But then, utilitarian alone fails to explain a variant of the trolley problem (commonly known as the *Footbridge Case*), as described below, which shares the same ending:

The initial setting is similar to that of the trolley problem, in which the runaway trolley is headed toward five people walking on the track, who are unable to get off the track in time. But in this variant, Ian is on the footbridge 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?

For this variant, as reported also in Hauser et al. (2007), most people tend to agree that it is *not* permissible for Ian to shove the man, even though shoving the heavy man would result in the same consequence of throwing the switch in the original trolley problem (also called the *Bystander Case*), viz., saving five people albeit killing one. The issue of these moral dilemmas (and other variants of the trolley problem, discussed in Section 8.1) is therefore addressing the permissibility of harming one or more individuals for the purpose of saving others.

The *Doctrine of Double Effect* has been referred to explain the consistency of judgments, shared by subjects from demographically diverse populations (across gender, ethnicity, religion, age, exposure to moral coursework, etc.) on a series of moral dilemmas (Hauser et al., 2007; Hauser, 2007; Mikhail, 2007). The Doctrine of Double Effect is first introduced by Thomas Aquinas in his discussion of the permissibility of self-defense (Aquinas, 1988). The current version of this principle all emphasize the permissibility of an action that causes a harm by distinguishing whether this harm is a mere *side-effect* of bringing about a good result, or rather an *intended means* to bringing about the same good end (McIntyre, 2004). According to the Doctrine of Double Effect, the former action is permissible, whereas the latter is impermissible. It provides a justification to distinguish the Bystander and the Footbridge cases. According to the Doctrine of Double Effect, shoving the heavy man, which causes his death, is impermissible because it is performed as an intended means to save the five. On the other hand, throwing the switch is permissible, because the death of the man on the side track is mere a side-effect of that action.

Another principle related to the Doctrine of Double Effect is the *Doctrine of Triple Effect* (Kamm, 2006). This principle refines the Doctrine of Double Effect particularly on the notion about harming someone as an intended means. That is, the Doctrine of Triple Effect distinguishes further between doing an action *in order* that an effect occurs and doing it *because* that effect will occur. The latter is a new category of action, which is not accounted for in the Doctrine of Double Effect. Though the Doctrine of Triple Effect also classifies the former as impermissible, it is more tolerant to the latter (the third effect), i.e., it treats as permissible those actions performed just *because* instrumental harm will occur.

The Doctrine of Triple Effect is proposed to accommodate another case of the trolley problem, the *Loop Case*, introduced by Thomson (1985), described below:

The initial setting is similar to that of the trolley problem, in which the runaway trolley is headed toward five people walking on the track, who are unable to get off the track in time. In this variant, Ned is standing next to a switch, which he can throw, that will temporarily turn the trolley onto a loop side track, which loops back towards the five. There is a heavy object on the side track. If the trolley hits the object, the object will slow the train down, giving the five people time to escape. The heavy object is a man. Ned can throw the switch, preventing the trolley from killing the five people, but killing the man. Or he can refrain from doing this, letting the five die. Is it morally permissible for Ned to throw the switch?

This case strikes most moral philosophers that throwing the switch to divert the trolley is permissible (Otsuka, 2008), though the Doctrine of Double Effect views that diverting the trolley in this Loop case as impermissible. Referring to the psychology experimental study by Hauser et al. (2007), 56% of its respondents judged that diverting the trolley in the Loop case is permissible. To this end, Kamm (2006) argues that the Doctrine of Triple Effect may provide the justification: it is permissible because it will hit the man, not in order to intentionally hit him.

But now, consider a variant of the Loop case, viz., the *Loop-Push Case* (cf. *Extra Push* Case in Kamm (2006)). In the Loop-Push case the looping side track is initially empty, but it has a footbridge over it with a heavy man standing on the footbridge. While throwing the switch diverts the trolley onto the side track, an ancillary action of pushing the heavy man can be performed in order to place him on the looping side track, so as to stop the runaway trolley going back to the main track.

How can this Loop-Push case be morally distinguished from the original Loop case? In the original Loop case, the man has already been on the looping side track when the trolley is diverted, which causes the death of the man. On the other hand, in the Loop-Push case, the suffered death of the man is not merely caused by diverting the trolley, as the looping side track is initially empty. Instead, the man dies as a consequence of a further action, i.e., pushing the man. This action is intentionally performed to place the man on the side track, for the trolley to hit the man, and eventually preventing the trolley from killing the five people.

Differently from the Loop case, in this Loop-Push case the Doctrine of Triple Effect agrees with the Doctrine Double Effect that it is impermissible to save the five, as the ancillary action of pushing the heavy man onto the looping the side track serves an intended mean, in order for his heavy body to stop the trolley.

3.1.2 Scanlonian Contractualism

Recently T. M. Scanlon, a professor of moral philosophy at Harvard University, has developed a distinctive view of moral reasoning called *contractualism* (Scanlon, 1998). It can be summarized as follows (Scanlon, 1998:p. 153):

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

The typical scope of Scanlonian contractualism is a domain of morality having to do with our duties to other people, referred to by Scanlon as "what we owe to each other", such as prohibitions against coercion, deception, harming and killing (like those exemplified by the trolley problem cases), rather than a broad moral criticism, such as premarital sex, homosexuality, etc., which are often considered immoral even they do not involve harming other people (Scanlon, 1998).

Scanlonian contractualism carries several features. First, it regards the importance of principles to provide reason for justifying an action to others, and emphasizes flexibility of moral principles. These principles may rule out some actions by ruling out the reasons on which they would be based, but they also leave wide room for interpretation, thus allowing exceptions. Indeed, as Dancy (2006) points out, a formulation of a principle may be expanded and become more complex, but decisive; what looks like a conflict between principles can really be viewed as a relation between incompletely specified principles, and the matter is resolved by a more complete specification of at least one of them.

Second, reasoning is an important aspect in contractualism. That is, moral decisions are not had by merely relying on internal observations, but through a process of careful assessment that is naturally called reasoning. In Scanlonian contractualism, the method of reasoning through which we arrive at a judgment of right and wrong is a primary concern, which is further fostered in Scanlon (2014), so as to explain a good reason to arrive at that judgment. More specifically, a judgment of right and wrong is about the adequacy of reasons for accepting or rejecting principles under certain conditions. This is clearly expressed within Scanlonian contractualism above, which seeks some common ground that others could not reasonably reject to, thus promoting the idea of justifiability of a moral judgment. From a different angle, morality in Scanlonian contractualism can be viewed as an attempt to arrive at (possibly defeasible) argumentative consensus or agreement. Of course, sometimes people may disagree in the consensual judgment, but

disagreement in judgments can be found in almost any area of inquiry, not just morality (Nagel, 2014).

In Scanlon (2008), moral permissibility is addressed through the so-called *deliberative employment* of moral principles, which is in line with Scanlonian contractualism. When principles are used to guide deliberation, the question of the permissibility of actions is answered by identifying the justified but defeasible argumentative considerations, and their exceptions. This employing of moral principles is based on a view that moral dilemmas, such as the trolley problem and other similar dilemmas, typically share the same structure. They concern general principles that in some cases admit exceptions, and they raise questions about when those exceptions apply. Providing such a structure, an action is determined impermissible through deliberative employment when there is no countervailing consideration that would justify an exception to the applied general principle.

3.2 The Dual-Process Model

In the recent years, there has been a number of psychological research about the *dual-process* that apparently forms the basis of our thinking, decision making and social judgments. In psychology, the dual-process commonly refers to two types of processes, called *Type 1* and *Type 2* (Evans, 2010; Stanovich, 2011), corresponding roughly to the familiar distinction between intuition and reflection. The Type 1 process is described as a fast and automatic process (it does not put a heavy load on central processing capacity), whereas Type 2 as a slow and controlled process. Further development of the dual-process theory has associated both types of processes with several attributes, e.g., experience-based decision making vs. consequential decision making, associative vs. rule-based, reactive vs. deliberative, etc. The detailed discussion of their distinctive attributes is beyond the scope of this thesis, but can be referred to Evans (2010); Evans and Stanovich (2013).

In Stanovich (2011), the modes of processing in these two types are represented as a tripartite model of mind: autonomous mind for Type 1 processing, whereas Type 2 consists of the algorithmic level of processing (the algorithmic mind) plus the reflective level of processing (the reflective mind). The autonomous mind implements short-leashed goals unless overridden by the algorithmic mind, where such an overriding is initiated by higher level control, viz., higher level goal states and epistemic thinking dispositions – both exist at the reflective level of processing. Stanovich (2011) also emphasizes the "cognitive decoupling" feature in Type 2 processing, which makes hypothetical thinking possible, by preventing our representations of the real world from becoming confused with representations of imaginary situations.

The dual-process model is evidenced by numerous psychological empirical tests, as discussed in Evans (2010) and Stanovich (2011). The role of the dual-process model in moral decision making is studied in Greene et al. (2004), by examining the neural activity of people responding to various moral dilemmas involving physically harmful behavior,

like those from the trolley problem cases. The experiments characterize each type in the dual-process with respect to applicable moral principles in these dilemmas. They found that a general principle favoring welfare-maximizing behaviors (utilitarian judgment in the Bystander case) appears to be supported by controlled cognitive processes (i.e., Type 2), whereas that prohibiting the use of harm as a means to a greater good (deontological judgment in the Footbridge case) appears to be part of the process that generates intuitive emotional responses (i.e., Type 1). When the utilitarian judgment is applied (such as in the Bystander case), the participants took longer time to make their responses. Moreover, the experiment suggests that in this case of utilitarian judgment, the Type 2 processing overrides the Type 1 coming from brain parts that produce emotion. Moral decision making is thus a product of complex interaction between these two types of the dual-process, even though each type apparently corresponds to a particular kind of moral judgment (Cushman et al., 2010).

Whereas Cushman et al. (2010) support the view that Type 2 is not associated with deontological (non-utilitarian) judgment, Mallon and Nichols (2010) stipulate that both reasoning on moral rules and emotion (which are associated to Type 2 and 1, respectively) work together to produce non-utilitarian judgment, like in the Footbridge case. This study is based on the view that moral judgment is supported by internally represented rules and reasoning about whether particular cases fall under those rules, even in deontological (non-utilitarian) judgment, where Type 1 dominates according to Cushman et al. (2010). It asserts that, though several studies demonstrate that people experience difficulty in justifying moral judgment generated by rapid and automatic processes (of Type 1), moral rules may still play an important role without the reasoning process being consciously accessible. Indeed, there has been no claim that Type 1 processing is non-computational, though the kind of rules that implement the Type 1 processing is not that of generally referred to in the Type 2 processing (Evans and Stanovich, 2013). The role of moral rules, even when the reasoning process about them is inaccessible consciously, thus provides an explanation that despite this difficulty, moral judgment driven by the affective system is able to mirror (and consistent with) a particular moral rule. For example, the deontological judgment in the Footbridge case is consistent with Doctrine of Double Effect, and so is the utilitarian judgment in the Bystander case.

3.3 Counterfactual Thinking in Moral Reasoning

Counterfactual literally means contrary to the facts. Counterfactuals are conjectures about what would have happened, had an alternative event occurred. They may be stated in a variety of linguistic constructs, e.g.:

- "If only I were taller ..."
- "I could have been a winner ..."
- "I would have passed, were it not for ..."

- "Even if . . . , the same consequence would have followed."

More generally, in psychology, counterfactuals are required to be in subjunctive mood rather than in indicative, which we follow in this thesis, as expressed by the following conditional statement:[2]

If the Antecedent had been true, then the Consequent would have been true.

Other requirements are that they must have false antecedent and they must be about particular past events (Woodward, 2011).

Counterfactual reasoning involves thoughts on what might have been, what could have happened, had some matter – action, outcome, etc. – been different in the past. It provides lessons for the future by virtue of contemplating alternatives. It permits thought debugging and supports a justification why different alternatives would have been worse or not better. It covers everyday experiences, like regret: "If only I had told her I love her!", "I should have studied harder". It may also triggers guilt responsibility, blame, and causation: "If only I had said something sooner, then I could have prevented the accident".

People typically reason about what they should or should not have done when they examine decisions in moral situation. It is therefore natural for them to engage counterfactual thoughts in such settings. As argued by Epstude and Roese (2008), the function of counterfactual thinking is not just limited to the evaluation process, but occurs also in the reflection one. Through evaluation, counterfactuals help correct wrong behavior in the past, thus guiding future moral decisions. Reflection, on the other hand, permits momentary experiential simulation of possible alternatives, thereby allowing careful consideration before a moral decision is made, and to subsequently justify it.

Counterfactual theories are very suggestive of a conceptual relationship to a form of debugging, namely in view of correcting moral blame, since people ascribe abnormal antecedents an increased causal power, and are also more likely to generate counterfactuals concerning abnormal antecedents. Two distinct processes can be identified when people engage in counterfactual thinking. For one, its frequently spontaneous triggers encompass bad outcomes and "close calls" (some harm that was close to happening). Second, such thinking comprises a process of finding antecedents which, if mutated, would prevent the bad outcome from arising. When people employ counterfactual thinking, they are especially prone to change abnormal antecedents, as opposed to normal ones. Following a bad outcome, people are likely to conceive of the counterfactual "if only [some abnormal thing] had not occurred, then the outcome would not have happened". For a review on this topic, see Roese (1997).

Morality and normality judgments typically correlate. Normality mediates morality with causation and blame judgments. McCloy and Byrne (2000) study the kind of counterfactual alternatives people tend to imagine, viz., those alternatives that can be controlled,

[2]From the language construct viewpoint, this subjunctive mood is commonly known as having the *third conditionals* form (Hewings, 2013).

in contemplating moral behaviors. The controllability in counterfactuals mediates between normality, blame and cause judgments. The importance of control, namely the possibility of counterfactual intervention, is highlighted in theories of blame that presume someone responsible only if they had some control of the outcome (Weiner, 1995).

There has been a number of studies, both in philosophy and psychology, on the relation between causation and counterfactuals. The *counterfactual process view* of causal reasoning (McCormack et al., 2011), for example, advocates counterfactual thinking as an essential part of the process involved in making causal judgments. This relation between causation and counterfactuals can be important for providing explanations in cases involving harm, which underlie people's moral cognition (Tetlock et al., 2007) and trigger other related questions, such as "Who is responsible?", "Who is to blame?", "Which punishment would be fair?", etc. In this thesis, we also explore the connection between causation and counterfactuals, focusing on agents' deliberate action, rather than on causation and counterfactuals in general. More specifically, our exploration of this topic links it to the Doctrines of Double Effect and Triple Effect and dilemmas involving harm, such as the trolley problem cases. Such cases have also been considered in psychology experimental studies concerning the role of gender and perspectives (first vs. third person perspectives) in counterfactual thinking in moral reasoning, see Migliore et al. (2014). The reader is referred to Collins et al. (2004) and Hoerl et al. (2011) for a more general and broad discussion on causation and counterfactuals.

3.4 Concluding Remarks

Our study on these moral facets leads us to some considerations, indicating how these moral facets relate to one another. They set the focus of our research and will guide our exploration for identifying appropriate LP concepts to represent and reason about them.

The relevance of the Doctrines of Double Effect (DDE) and Triple Effect (DTE) is constrained in our research by their applicability to address moral permissibility. This research does not intend to cover moral facets with various deontic categories (obligation, prohibition, and permissibility) nor to formalize them using some deontic logic formalisms. Such research line has been pursued elsewhere, e.g., in Powers (2006) and Bringsjord et al. (2006). Moreover, our focus on this moral principles is anchored to their relevant moral examples, viz., the various cases of the trolley problem, which have been subject of experiments, whose results are readily available in the literature for validating our models.

Scanlonian contractualism concerns the *process* of moral decision making, which can sufficiently be demonstrated by an informal consensual argumentation, namely in justifying some actions or decisions in moral cases. Following Scanlonian contractualism, moral permissibility is addressed by identifying the justified but defeasible argumentative considerations. These considerations are based on moral principles, which in some cases admit exceptions. Indeed, such moral principles may be captured by the DDE and DTE to some extent. Each of these principles in itself admit some exceptions, as demonstrated by

different judgments, e.g., in the Bystander and Footbridge cases with DDE as their referred moral principle. Furthermore, the third effect in DTE can be viewed as an exception to DDE, making DTE in some cases an exception to the DDE-impermissibility. It therefore seems appropriate to consider them in illustrating the argumentation process in justifying some moral decisions.

The dual-process model refers to an interaction between reactive and deliberative psychological processes in decision making. Its concept has been known close to computational procedure (Kowalski, 2011; Stanovich, 2011). It also has been empirically studied in psychology using, amongst others, the Bystander and Footbridge cases of the trolley problem, as reported by Greene et al. (2004) and Cushman et al. (2010), in Section 3.2. Their results, viz., concerning the different kind of processes in deontological and utilitarian judgments, will be taken as reference for finding a close LP representation to moral decision making. Another important reference for representing the interaction between the two processes is the attributes that are often associated with them, such as fast vs. slow, reactive vs. deliberative, etc.

Finally, besides exploring the connection between counterfactual and causation, and linking it to the Doctrine of Double and Triple Effect (as described in the previous section), we may look into its further application for justifying permissibility through argumentation processes à la Scanlonian contractualism.

All considerations in this concluding remarks are addressed in the following chapter.

CHAPTER 4

REPRESENTING MORALITY IN LOGIC PROGRAMMING

We start with a general Logic Programming background and notation used throughout this thesis, in Section 4.1, following the notation from Alferes and Pereira (1996). The subsequent sections enumerate features in LP-based reasoning considered in this thesis, and discuss their appropriateness in representing various issues of moral facets elaborated in Chapter 3.

4.1 Preliminaries

By an alphabet $\mathcal{A}$ of a language $\mathcal{L}$ we mean a countable disjoint set of constants, function symbols, and predicate symbols. Moreover, an alphabet is assumed to contain a countable set of variable symbols. We use the underscore symbol (_) to specifically denote an anonymous variable. A term over $\mathcal{A}$ is defined recursively as either a variable, a constant or an expression of the form $f(t_1, \ldots, t_n)$, where f is a function symbol of $\mathcal{A}$, and t_is are terms. An *atom* over $\mathcal{A}$ is an expression of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol of $\mathcal{A}$, and t_is are terms. In this thesis, we write p/n to denote the predicate symbol p having arity n. A *literal* is either an atom a or its negation *not* a. Literals of the latter form is called default literals.

A term (respectively, atom and literal) is *ground* if it does not contain variables. The set of all ground terms (respectively, ground atoms) of $\mathcal{A}$ is called the Herbrand universe (respectively, Herbrand base) of $\mathcal{A}$.

Definition 1 (Logic Program) *A* (normal) logic program *is a countable set of rules of the form:*

$$H \leftarrow L_1, \ldots, L_m$$

where H is an atom, $m \geq 0$, and L_is ($1 \leq i \leq m$) are literals.[1]

The comma operator in rules is read as conjunction. A normal logic program is called *definite* if none of its rules contains default literals. Following the standard convention, rules of the form $H \leftarrow$ are alternatively written as H. A rule of this form is called a *fact*.

The alphabet $\mathcal{A}$ used to write program P is assumed to precisely comprise all the constants, the function and predicate symbols that explicitly appear in P. By Herbrand universe (respectively, base) of P we mean the Herbrand universe (respectively, base) of $\mathcal{A}$. We denote the Herbrand base of P by $\mathcal{H}_P$. By a ground logic program we mean the set of ground rules obtained from P by substituting in all possible ways each of the variables in P by elements of its Herbrand universe.

We define next two- and three-value Herbrand interpretations and models of logic programs.[2]

Let F be a set of atoms, $F = \{a_1, \ldots, a_n\}$. By *not* F we mean the set $\{not\ a_1, \ldots, not\ a_n\}$.

Definition 2 (Two-valued Interpretation) *A* two-valued interpretation *I of a logic program P is a set of literals*

$$I = T \cup not\ F$$

such that $T \cup F = \mathcal{H}_P$ and $T \cap F = \emptyset$.

The set T (respectively, F) is the set of atoms that are true (respectively, false) in I. The interpretation I is said two-valued because the truth value true or false is assigned to precisely one ground atom, viz., $T \cup F = \mathcal{H}_P$ and $T \cap F = \emptyset$.

Alternatively, the three-valued interpretation is defined below. It permits representing incomplete knowledge, where some atoms are neither true nor false, but rather undefined.

Definition 3 (Three-valued Interpretation) *A* three-valued interpretation *I of a logic program P is a set of literals*

$$I = T \cup not\ F$$

such that $T \subseteq \mathcal{H}_P$, $F \subseteq \mathcal{H}_P$ and $T \cap F = \emptyset$.

In a three-valued interpretation, the set T (respectively, F) is the set of atoms that are true (respectively, false) in I, and the truth value of the remaining atoms is undefined. Clearly, the two-valued interpretation is a special case of the three-valued one, for which $T \cup F = \mathcal{H}_P$ is additionally imposed.

[1] In the sequel, unless otherwise specified, we generally write *logic programs* to refer to *normal logic programs*.

[2] In the sequel, we simply write *interpretations* and *models* to refer to *Herbrand interpretations* and *Herbrand models*, respectively.

We may view an interpretation I of a program P as a a function $I : \mathcal{H}_P \to \mathcal{V}$, where $\mathcal{V} = \{0, 0.5, 1\}$, defined by:

$$I(A) = \begin{cases} 0 & \text{if } \textit{not } A \in I \\ 1 & \text{if } A \in I \\ 0.5 & \text{otherwise} \end{cases}$$

Clearly, for two-valued interpretations there is no atom A such that $I(A) = 0.5$.

Models are defined as usual, and based on a truth valuation function.

Definition 4 (Truth Valuation) *If I is an interpretation, the* truth valuation *$\hat{I}$ corresponding to I is a function $\hat{I} : \mathcal{F} \to \mathcal{V}$, where $\mathcal{F}$ is the set of ground literals, conjunctions of literals, and rules formed over the language. It is defined as follows:*

- *If L is a ground atom, then $\hat{I}(L) = I(L)$.*
- *If L is a default literal, i.e., $L = not\ A$, then $\hat{I}(L) = 1 - \hat{I}(A)$.*
- *If S and T are conjunctions of literals, then $\hat{I}((S, T)) = min(\hat{I}(S), \hat{I}(T))$.*
- *If $H \leftarrow B$ is a rule, where B is a conjunction of literals, then:*

$$\hat{I}(H \leftarrow B) = \begin{cases} 1 & \textit{if } \hat{I}(B) \leq \hat{I}(H) \\ 0 & \textit{otherwise} \end{cases}$$

For any $F \in \mathcal{F}$, the values 0, 0.5 and 1 of $\hat{I}(F)$ correspond to the truth values false, undefined *and* true, *respectively. We write $I \models F$, for $F \in \mathcal{F}$, iff $\hat{I}(F) = 1$.*

Definition 5 (Model) *A interpretation I is called a (two-valued or three-valued)* model *of a program P iff for every ground instance $H \leftarrow B$ of a rule in program P we have $\hat{I}(H \leftarrow B) = 1$.*

We define some orderings among interpretations and models as follows.

Definition 6 (Classical Ordering (Przymusinski, 1989a)) *If I and J are two interpretations then we say that $I \preceq J$ if $I(A) \leq J(A)$ for any ground atom A. If $\mathcal{I}$ is a collection of interpretations, then an interpretation $I \in \mathcal{I}$ is called* minimal *in $\mathcal{I}$ if there is no interpretation $J \in \mathcal{I}$ such that $J \preceq I$ and $J \neq I$. An interpretation I is called* least *in $\mathcal{I}$ if $I \preceq J$, for any other interpretation $J \in \mathcal{I}$. A model M is called* minimal *(respectively,* least*) if it is minimal (respectively, least) among all models of P.*

Definition 7 (Fitting Ordering (Fitting, 1985)) *If I and J are two interpretations then we say that $I \preceq_F J$ iff $I \subseteq J$. If $\mathcal{I}$ is a collection of interpretations, then an interpretation $I \in \mathcal{I}$ is called* F-minimal *in $\mathcal{I}$ if there is no interpretation $J \in \mathcal{I}$ such that $J \preceq_F I$ and $J \neq I$. An interpretation I is called* F-least *in $\mathcal{I}$ if $I \preceq_F J$, for any other interpretation $J \in \mathcal{I}$. A model M is called* F-minimal *(respectively,* F-least*) if it is F-minimal (respectively, F-least) among all models of P.*

Note that the classical ordering is related with the *degree of truth* of their atoms, whereas the Fitting ordering is related with the *degree of information*. Under the latter ordering, the undefined value is less than both values true and false, providing that true and false being incompatible.

In Emden and Kowalski (1976), it is shown that every definite program has a unique least model, which determines the so-called *least model semantics* of a definite program. Other semantics for more general programs, allowing default literals in the body of a rule, have been proposed. In Gelfond and Lifschitz (1988), *Stable Model Semantics* is introduced. Informally, when one assumes true some set of (hypothetical) default literals, and false all the others, some consequences follow according to the semantics of definite programs. If the consequences completely corroborate the hypotheses made, then they form a stable model. We first introduce the *Gelfond-Lifschitz* operator Γ that operates on two-valued interpretations of a program P.

Definition 8 (Gelfond-Lifschitz Operator) *Let P be a logic program and I be its two-valued interpretation. The* GL-transformation of P modulo I *is the program $\frac{P}{I}$ obtained from P by performing the following operations:*

- *Remove from P all rules which contain a default literal $L = not\ A$ such that $\hat{I}(L) = 0$;*
- *Remove from all remaining rules those default literals $L = not\ A$ which satisfy $\hat{I}(L) = 1$.*

Since the resulting program $\frac{P}{I}$ is a definite program, it has a unique least model J. We define $\Gamma(I) = J$.

In Gelfond and Lifschitz (1988) it is shown that fixed points of the Gelfond-Lifschitz operator Γ for a program P are minimal models of P.

Definition 9 (Stable Model Semantics) *A two-valued interpretation I of a logic program P is a* stable model *of P if $\Gamma(I) = I$.*

Example 1 *The program P:*

$$\begin{aligned} a &\leftarrow not\ b.\\ b &\leftarrow not\ a.\\ c &\leftarrow not\ d.\\ d &\leftarrow not\ e.\\ p &\leftarrow a.\\ p &\leftarrow b. \end{aligned}$$

has two stable models: $I_1 = \{a, d, p, not\ b, not\ c, not\ e\}$ and $I_2 = \{b, d, p, not\ a, not\ c, not\ e\}$. One can verify that $\frac{P}{I_1}$ is the program:

$$\begin{aligned} a &\leftarrow .\\ d &\leftarrow .\\ p &\leftarrow a.\\ p &\leftarrow b. \end{aligned}$$

Therefore, $\Gamma(I_1) = I_1$.

Similarly for I_2, *the program* $\frac{P}{I_2}$ *is:*

$$\begin{array}{lll} b & \leftarrow & . \\ d & \leftarrow & . \\ p & \leftarrow & a. \\ p & \leftarrow & b. \end{array}$$

Therefore, $\Gamma(I_2) = I_2$.

Despite its advantages, that it provides semantics for more general programs than its predecessors and is closely related to autoepistemic logic and default theory (see Gelfond (1987) and Bidoit and Froidevaux (1991)), Stable Model Semantics has some drawbacks. Some programs may have no stable models, e.g., the program $p \leftarrow not\ p$. Even for programs with stable models, their semantics do not always lead to the expected intended semantics (see Alferes and Pereira (1996) for a discussion).

The *Well-Founded Semantics* is introduced in Gelder et al. (1991), addressing the difficulties encountered with the Stable Model Semantics. It has been shown in Przymusinski (1989b) that the Well-Founded Semantics is also equivalent to major formalizations of non-monotonic reasoning.

The Well-Founded Semantics can be viewed as *three-valued Stable Model Semantics* (Przymusinska and Przymusinski, 1990). In order to formalize the notion of three-valued stable models, the language of programs is expanded with the additional propositional constant **u** with the property of being undefined in every interpretation. It is therefore assumed that every interpretation I satisfies:

$$\hat{I}(\mathbf{u}) = \hat{I}(not\ \mathbf{u}) = 0.5$$

A *non-negative* program is a program whose rules' bodies are either atoms or **u**. It is proven in Przymusinska and Przymusinski (1990) that every non-negative logic program has a unique least three-valued model.

The next definition extends the Gelfond-Lifschitz operator Γ to a three-valued operator Γ^*.

Definition 10 (Γ^*-operator) *Let P be a logic program and I be its three-valued interpretation. The* extended GL-transformation of P modulo I *is the program* $\frac{P}{I}$ *obtained from P by performing the following operations:*

- *Remove from P all rules which contain a default literal $L = not\ A$ such that $\hat{I}(L) = 0$;*
- *Replace in the remaining rules of P those default literals $L = not\ A$ which satisfy $\hat{I}(L) = 0.5$ by* ***u****;*
- *Remove from all remaining rules those default literals $L = not\ A$ which satisfy $\hat{I}(L) = 1$.*

Since the resulting program $\frac{P}{I}$ *is non-negative, it has a unique three-valued least model* J*. We define* $\Gamma^*(I) = J$*.*

Definition 11 (Well-Founded Semantics) *A three-valued interpretation* I *of a logic program* P *is a three-valued stable model of* P *if* $\Gamma^*(I) = I$*. The* Well-Founded Semantics *of* P *is determined by the unique F-least three-valued stable model of* P*, and can be obtained by the bottom-up iteration of* Γ^* *starting from the empty interpretation.*

Example 2 *Recall the program in Example 1. Let* $I_0 = \emptyset$ *be the empty interpretation.*

- *The least three-valued model of* $\frac{P}{I_0}$*:*

$$\begin{array}{lll} a & \leftarrow & \boldsymbol{u}. \\ b & \leftarrow & \boldsymbol{u}. \\ c & \leftarrow & \boldsymbol{u}. \\ d & \leftarrow & \boldsymbol{u}. \\ p & \leftarrow & a. \\ p & \leftarrow & b. \end{array}$$

  is $\Gamma^*(I_0) = \{not\ e\}$*.*

- *Let* $I_1 = \Gamma^*(I_0)$*. The least three-valued model of* $\frac{P}{I_1}$*:*

$$\begin{array}{lll} a & \leftarrow & \boldsymbol{u}. \\ b & \leftarrow & \boldsymbol{u}. \\ c & \leftarrow & \boldsymbol{u}. \\ d & \leftarrow & . \\ p & \leftarrow & a. \\ p & \leftarrow & b. \end{array}$$

  is $\Gamma^*(I_1) = \{d, not\ e\}$*.*

- *Let* $I_2 = \Gamma^*(I_1)$*. The least three-valued model of* $\frac{P}{I_2}$*:*

$$\begin{array}{lll} a & \leftarrow & \boldsymbol{u}. \\ b & \leftarrow & \boldsymbol{u}. \\ d & \leftarrow & . \\ p & \leftarrow & a. \\ p & \leftarrow & b. \end{array}$$

  is $\Gamma^*(I_2) = \{d, not\ c, not\ e\}$*.*

- *Let* $I_3 = \Gamma^*(I_2)$*. The least three-valued model of* $\frac{P}{I_3}$*:*

$$\begin{array}{lll} a & \leftarrow & \boldsymbol{u}. \\ b & \leftarrow & \boldsymbol{u}. \\ d & \leftarrow & . \\ p & \leftarrow & a. \\ p & \leftarrow & b. \end{array}$$

is $\Gamma^*(I_3) = \{d, not\ c, not\ e\}$.

Therefore, the well-founded model of P is $I_3 = \{d, not\ c, not\ e\}$, where d is true, c and e are both false, and a, b and p are undefined.

In the sequel, we write the well-founded model of program P as $WFM(P)$.

4.2 Abduction

The notion of abduction is first introduced by Peirce (1932), and is characterized as "a step of adopting a hypothesis as being suggested by the facts".

Abduction consists of reasoning where one chooses from available hypotheses those that best explain the observed evidence, in some preferred sense. Abduction in LP is realized by extending LP with abductive hypotheses, called abducibles. An *abducible* is an atom Ab or its negation Ab^* (syntactically an atom, but denoting literal $not\ Ab$), named *positive* and *negative abducibles*, respectively, whose truth value is not initially assumed. The *negation complement of an abducible* A is denoted by $compl_{\mathcal{AB}}(A)$, where the complement of a positive abducible Ab and its negation Ab^* is defined as $compl_{\mathcal{AB}}(Ab) = Ab^*$ and $compl_{\mathcal{AB}}(Ab^*) = Ab$, respectively.

We next define an abductive framework in LP (Kakas et al., 1992), which includes integrity constraints for restricting abduction. The definitions in this section are adapted from those of Alferes et al. (2004a).

Definition 12 (Integrity Constraint) *An* integrity constraint *is a rule in the form of a denial:*

$$\bot \leftarrow L_1, \ldots, L_m.$$

where $\bot/0$ *is a reserved predicate symbol in* $\mathcal{L}$, $m \geq 1$, *and* L_i*s* ($1 \leq i \leq m$) *are literals.*

Definition 13 (Abductive Framework) *An* abductive framework *is a triple* $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$, *where* $\mathcal{AB}$ *is the set of abducible predicates (and their corresponding arity),* P *is a logic program over* $\mathcal{L} \setminus \{\bot\}$ *such that there is no rule in* P *whose head is an abducible formed by a predicate in* $\mathcal{AB}$, *and* $\mathcal{IC}$ *is a set of integrity constraints.*

Given an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$, we write $\mathcal{AB}_{gL}$ to denote a finite set of ground abducibles formed over the set $\mathcal{AB}$.[3] In particular, $\mathcal{AB}_{gL} = \mathcal{AB}$ if the abducible predicates in $\mathcal{AB}$ are just propositional (nullary predicates).

Definition 14 (Abductive Scenario) *Let* F *be an abductive framework* $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$. *An* abductive scenario *of* F *is a tuple* $\langle P, \mathcal{AB}, \mathcal{S}, \mathcal{IC} \rangle$, *where* $\mathcal{S} \subseteq \mathcal{AB}_{gL}$ *and there is no* $A \in \mathcal{S}$ *such that* $compl_{\mathcal{AB}}(A) \in \mathcal{S}$, *i.e.,* $\mathcal{S}$ *is consistent.*

[3] In practice, the body of a program rule may contain non-ground abducibles, but they have to be ground when abduced.

The consistency of an abductive scenario can be imposed by an integrity constraint $\bot \leftarrow Ab, Ab^*$.

Let observation O be a set of literals, analogous to a query in LP. Abducing an explanation for O amounts to finding consistent abductive solutions to a goal, whilst satisfying the integrity constraints, where abductive solutions consist in the semantics obtained by replacing in P the abducibles in $\mathcal{S}$ by their truth value. We define formally abductive solutions under the Well-Founded Semantics below.

Given an abductive scenario $\langle P, \mathcal{AB}, \mathcal{S}, \mathcal{IC} \rangle$ of an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$, we first define $P_{\mathcal{S}}$ as the smallest set of rules that contains for each $A \in \mathcal{AB}_{gL}$, the fact A if $A \in \mathcal{S}$; and $A \leftarrow \mathbf{u}$ otherwise. Alternatively, and obviously equivalent, instead of adding to $P_{\mathcal{S}}$ the rule $A \leftarrow \mathbf{u}$, one may simply replace the corresponding A with $\mathbf{u}$ both in P and $\mathcal{IC}$.

Definition 15 (Abductive Solution) *Let F=$\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ and $\langle P, \mathcal{AB}, \mathcal{S}, \mathcal{IC} \rangle$ be an abductive scenario of F. The consistent set of abducibles $\mathcal{S}$ is an* abductive solution *to F if $\bot$ is* **false** *in $M_s = WFM(P \cup P_{\mathcal{S}} \cup \mathcal{IC})$. We say that $\mathcal{S}$ is an* abductive solution for query Q *if Q is* **true** *in M_s, written $M_s \models Q$.*

Abduction in LP can be accomplished by a top-down query-oriented procedure for finding a query solution by need. The solution's abducibles are leaves in its procedural query-rooted call-graph, i.e., the graph is recursively generated by the procedure calls from literals in bodies of rules to heads of rules, and thence to the literals in a rule's body. The correctness of this top-down computation requires the underlying semantics to be relevant, as it avoids computing a whole model (to warrant its existence) in finding an answer to a query. Instead, it suffices to use only the rules relevant to the query – those in its procedural call-graph – to find its truth value. The Well-Founded Semantics enjoys this relevancy property, i.e., it permits finding only relevant abducibles and their truth value via the aforementioned top-down query-oriented procedure. Those abducibles not mentioned in the solution are indifferent to the query.

Representing Moral Facets by Abduction

The basic role of abduction pertains to its applicability for decision making, where abducibles are used for representing available decisions in moral dilemmas. For example, in the trolley problem, one can introduce abducibles to represent decisions like 'diverting the trolley', 'shoving the heavy man', etc., depending on the considered cases. They are abduced to satisfy a given query and integrity constraints, which reflect moral considerations in a modeled dilemma.

Other roles of abduction in representing the identified moral facets are as follows:

- With respect to moral permissibility, integrity constraints can be used for ruling out impermissible actions according to the followed moral principle. For example, when representing the Doctrine of Double Effect, integrity constraints are used for

excluding those actions corresponding to intended harms for a greater good. It can therefore capture the deontological judgment with respect to the Doctrine of Double Effect. That is, integrity constraints can be applied *a priori* for ruling out intended harming actions, regardless how good their consequences are. On the other hand, utilitarian judgments, such as in the Bystander case, require weighing the consequences of decisions in order to arrive at a right decision that maximizing the best consequences. This is more appropriately addressed by other feature, as discussed in Section 4.3.

- In counterfactual reasoning, whose applications concerning moral permissibility are explored in this thesis, hypothetically conjecturing an alternative event requires the "other things being equal" assumption for fixing the background context of the counterfactual being evaluated. Abduction is important for providing such "other things being equal" background context. To this end, it hypothesizes incomplete information about the setting (the exogenous variables) by providing an explanation to the factual observations. This will be discussed in Chapter 6.

Related to abduction is the concept of inspection points (Pereira et al., 2013), where side-effects in abduction are examined. This concept is declaratively construed with a program transformation, and procedurally constructed by 'meta-abducing' a specific abducible literal $abduced(A)$ whose function is only checking that its corresponding abducible A is indeed already abduced elsewhere. Therefore, the consequence of the action that triggers this 'meta-abducing' is merely a side-effect. In this thesis, the use of inspection points is only illustrated as an alternative to counterfactual reasoning in representing the Doctrine of Double Effect, by distinguishing between an instrumental cause and a mere side-effect.

For related use of inspection points to express contextual side effects and other variants of contextual abductive explanations, and their applications for modeling belief-bias effect in psychology (Evans, 2012; Evans et al., 1983), the reader is referred to Pereira et al. (2014).

4.3 Preferences over Abductive Scenarios

In abduction it is desirable to generate only abductive explanations relevant for the problem at hand. In Dell'Acqua and Pereira (2007), abducibles in an abductive framework are selectively assumed by introducing rules encoding domain specific information about which particular assumptions are relevant and considered in a specific situation, namely which can be instrumentally used and preferred. For this purpose, the notion of *expectation* is employed to express preconditions for an expectation of an abducible A (or its contrary), as expressed by rule $expect/1$ (or $expect\_not/1$, respectively) below:

$$\begin{aligned} expect(A) &\leftarrow L_1, \ldots, L_m. \\ expect\_not(A) &\leftarrow L_1, \ldots, L_n. \end{aligned}$$

Using this notion of expectation, an abducible A is only abduced if there is an expectation for it, and there is no expectation to the contrary. In this case, we say that the abducible A

is *considered*. We discuss in Chapter 7 two possible rules defining $consider(A)$, in terms of $expect(A)$ and $expect\_not(A)$. They particularly differ in the technique employed for abducing A: one realizes abduction using an even loop over negation under Stable Model Semantics, whereas the other concerns abduction in Well-Founded Semantics (see Definition 15).

While the above concept of expectation constrains relevant abducibles *a priori* with respect to the agent's actual situation, other preferences can be enacted by examining the consequences of the considered abducibles (Pereira et al., 2013). These consequences are contained in the abductive stable models of the considered abducibles.

Definition 16 (Abductive Stable Model) *Let $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ be an abductive framework and $\Delta \subseteq \mathcal{AB}_{gL}$ be a set of abducibles. An interpretation M is an* abductive stable model of Δ *if M is a stable model of $P \cup \Delta$.*

Because stable models are two-valued, each abducible or its negation complement must be abduced, subject to consistency.

Since these consequences are contained in abductive stable models, such preferences can only be enacted *a posteriori*, viz., only after considered abducibles and their abductive stable models are computed. In evaluating consequences, a posteriori preferences can be based on quantitative measures (e.g., by utility functions) or qualitative ones (e.g., by enforcing some property over consequences to hold).

Representing Moral Facets by Preferences

In this thesis, a posteriori preferences are appropriate for capturing utilitarian judgment that favors welfare-maximizing behaviors. More specifically, the combined use of a priori integrity constraints and a posteriori preferences reflects an interaction of two different processes in terms of the dual-process model. Whereas a priori integrity constraints can be viewed as a mechanism to generate immediate responses in deontological judgment, reasoning with a posteriori preferences can be viewed as a form of controlled cognitive processes in utilitarian judgment. The latter is evidently a more involved process of reasoning: after excluding those abducibles that have been ruled out a priori by the integrity constraints, the consequences of the considered abducibles have first to be computed, and only then are they evaluated to prefer the solution affording the greater good.

4.4 Probabilistic LP

Moral reasoning is typically performed upon conceptual knowledge of the actions. But it is often the case 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. When

such uncertainty is expressed in probability values, it is relevant to probabilistically reason about actions.

A number of research has been done for integrating probability in LP, which results in a paradigm called Probabilistic LP. Many languages have been proposed in Probabilistic LP, amongst them: Independent Choice Logic (Poole, 1997), PRISM (Sato, 1995), Logic Programs with Annotated Disjunctions (Vennekens et al., 2004), ProbLog (Raedt et al., 2007), P-log (Baral et al., 2009) and PITA (Riguzzi and Swift, 2011). These languages share similarities in their semantics, viz., using various forms of the *distribution semantics* (Sato, 1995). Under the distribution semantics, a probabilistic logic program defines a probability distribution over normal logic programs (called *worlds*). The distribution is extended to a joint probability distribution over worlds and interpretations, from which the probability of a query can be obtained. The reader is referred to Riguzzi and Swift (2014) for a survey on various Probabilistic LP under the distribution semantics.

Representing Moral Facets by Probabilistic LP

In this thesis, Probabilistic LP is employed as part of abduction in order to allow abducing moral decisions under uncertainty, in particular for reasoning about actions with respect to the availability of observed evidences and their attending truth value. This is relevant in moral jurisprudence, e.g., in courts, where jurors are required to proffer rulings beyond reasonable doubt based on available evidences.

The use of Probabilistic LP in court rulings illustrates a form of deliberative employment (of Scanlonian contractualism), where permissibility of actions is addressed through justified but defeasible argumentative considerations. This stance can be captured by viewing the standard probability of proof beyond reasonable doubt (for an example of such probability, see Newman (2006)) as a common ground for the verdict of guilty to be qualified as 'beyond reasonable doubt'. Argumentation may subsequently take place through presentation of other evidences with diverse level of uncertainty (these evidences are presented via LP updating, to be described in Section 4.5) as a consideration to justify exceptions. Whether such an evidence is accepted as a justification (by defeating the formerly presented evidence) depends on its influence on the probability of action, which in turn determines its permissibility and thereby the verdict. In other words, it depends on whether this probability is still within the agreed standard of proof beyond reasonable doubt, given that moral permissibility of actions is couched in court terms as verdicts about guilt.

4.5 LP Updating

Concomitantly to abduction an agent may learn new information from the external world or update itself internally of its own accord in order to pursue its present goal. It is therefore natural to accommodate LP abduction with updating.

LP updating allows Logic Programming to represent dynamic knowledge, i.e., knowledge that evolves over time, where updating is not only performed upon the extensional part (facts of a program), but may also take place on the intensional part (rules of a program). LP updating has been extensively studied, leading to the development of LP updating semantics (Alferes et al., 2000; Alferes et al., 2005; Eiter et al., 2002; Leite, 2003), a Dynamic Logic Programming framework for LP updating (Alferes et al., 2000), and languages of LP updating (Alferes et al., 2002a; Leite, 2003).

In this thesis, the application of LP updating is restricted to updating fluents only. Nevertheless, this restriction is sufficient for dealing with morality facets considered in this thesis.

Representing Moral Facets by LP Updating

LP updating permits establishing a concrete exception to a moral principle. It thus facilitates modeling Scanlon's deliberative employment for justifying permissible actions, which involves defeasible argumentation and consideration through exceptions. The role of LP updating is particularly relevant in the following moral cases:

- The knowledge of jurors in a court case, as discussed in the previous section, may be updated with new evidences, which may defeat former ones, and consequently may influence the verdict (depending on the agreed probability standard of the verdict).
- In the trolley problem, the knowledge of an agent may be updated with new information, allowing a scenario to be constructed incrementally. It thus permits demonstrating how an argumentation takes place to defend the permissibility of the agent's action in the presence of the newly obtained information.
- The agent may also adopt a new (possibly overriding) moral rule on top of those an agent currently follows, where the updating moral rule can be viewed as an exception to the current one. Such updating is necessary when the currently followed moral rule has to be revised, or qualified by an overriding exception, in the light of the situation faced by the agent.

In another role, together with abduction, LP updating forms a procedure to evaluate the validity of counterfactuals:

- Whereas abduction in the procedure provides some explanations to the factual observation, LP updating updates the causal model with a preferred explanation. It thus fixes the "other things being equal" background context of the counterfactual being evaluated into the causal model.
- LP updating also helps establish an intervention to the causal model, which is realized via a hypothetical update for representing a hypothetical alternative former event or action.

4.6 LP Counterfactuals

Counterfactual thinking in moral reasoning has been investigated particularly via psychology experiments (see, e.g., McCloy and Byrne (2000), Migliore et al. (2014) and Epstude and Roese (2008)), but it has only been limitedly explored in machine ethics. In this thesis, the application of counterfactual reasoning to machine ethics is fostered, while also bringing counterfactuals to a wider context of the aforementioned well-developed LP-based reasoning features by introducing a pure non-probabilistic LP-based approach for evaluating the validity of counterfactuals. It is therefore appropriate for cases when probabilities are not known or needed.

The newly introduced approach complements existing probabilistic approaches of counterfactual reasoning in LP – the latter approaches have been studied elsewhere (Baral and Hunsaker, 2007; Vennekens et al., 2010) – by formulating counterfactual reasoning using abduction and updating.

Representing Moral Facets by Counterfactuals

We specifically employ counterfactual reasoning to examine moral permissibility of actions according to the Doctrines of Double Effect and Triple Effect. This is achieved by distinguishing between a *cause* and a *side-effect* as a result of performing an action in order to achieve a goal. This distinction is important in explaining the permissibility of an action, both in the Doctrines of Double Effect and Triple Effect. While the issue of moral permissibility with respect to these two doctrines can be addressed using abduction with integrity constraints and a posteriori preferences (as described in Sections 4.2 and 4.3), the use of counterfactuals provides a different approach, by means of a general counterfactual conditional form, to examine moral permissibility according to these doctrines. Furthermore, counterfactual reasoning may take place in justifying moral permissibility:

- In the form of *compound counterfactuals*, "Had I known what I know today, then if I were to have done otherwise, something preferred would have followed", for justifying with hindsight what was done in the past, in the absence of current knowledge.

- In the spirit of Scanlonian contractualism, conceptual counterfactual queries can be employed for providing a justified, but defeasible, exception to permissibility of actions.

4.7 Tabling

Tabling affords solutions reuse, rather than recomputing them, by keeping in tables subgoals and their answers obtained from query evaluation. It is now supported by a number of Prolog systems (to different extent of features), such as XSB Prolog, YAP

Prolog, B-Prolog, Ciao, Mercury, and ALS Prolog. The simple idea of tabling has profound consequences (Swift and Warren, 2012):

- Tabling ensures termination of programs with the *bounded term-size property*, viz., those programs where the sizes of subgoals and answers produced during an evaluation are less than some fixed number.
- Tabling can be extended to evaluate programs with negation according to the Well-Founded Semantics.
- For queries to wide classes of programs, such as datalog programs with negation, tabling can achieve the optimal complexity for query evaluation.
- Tabling integrates closely with Prolog, so that Prolog's familiar programming environment can be used, and no other language is required to build complete systems.

For a tutorial on tabling, the reader is referred to Chapter 5 of Swift et al. (2015).

Representing Moral Facets by Tabling

This idea of solution reuse suggests that tabling is appropriate for an interaction of the controlled and the intuitive processes, and therefore capturing the dual-process model in moral decision-making. That is, while the controlled part of moral decision-making requires deliberative reasoning (via goal-oriented abductive reasoning), it may also rely on the tabling mechanisms, at Prolog system level, for readily obtaining solutions from tables rather than deliberatively recomputing them. The availability of such tabled solutions, managed at system level, is in line with the view of the low-level intuitive process of the dual-process model, which permits a rapid and automatic moral judgment.

In order to realize the above interaction in the dual-process model, we introduce in this thesis two engineering techniques for taking the benefit of tabling into two reasoning features, viz., tabling abductive solutions in contextual abduction and incremental tabling of fluents for (dual-process like) bottom-up LP updating, as well as in their integration. The resulting system is then employed to conceptually demonstrate the dual-process model in moral permissibility, where the roles of tabling in this aspect are as follows:

- As reported in Cushman et al. (2010), the dual-process model associates the intuitive process with the deontological judgment in moral dilemmas like those of the trolley problem. Given that abductive solutions represent those decisions that have been abduced according to a specific deontological moral principle (e.g., the Doctrine of Double Effect), tabling in abduction allows an agent to immediately deliver an action in a compatible context, without repeating the same deliberative reasoning. The consistency of moral decision can therefore be maintained, following that specific moral principle (as if it is obtained through deliberative reasoning), even though this decision is only directly retrieved from the table (via tabling abductive solutions).

- While the benefit of solution reuse in tabling captures the low-level and reactive parts of the dual-process model, the interaction between the deliberative and the reactive processes of this model can be demonstrated by the combination of tabling in abduction and in updating. In particular, incremental tabling may trigger an automatic (at system level) bottom-up updates propagation, where such propagation is driven by an actual abductive goal query. A scenario would be that an agent obtains some new information while making a moral decision (satisfying a given goal). In such a dynamic moral situation, where the new information is propagated by incremental tabling through its knowledge base, the agent may later be required to relaunch its previous goal or to additionally achieve some new goals in the presence of this new information and its consequences. As before, while achieving the new goals requires an extra deliberative reasoning, the decisions that have been abduced for former goals can immediately be retrieved from the table, and can subsequently be reused in the deliberative reasoning for the new goals, via contextual abduction.

4.8 Concluding Remarks

While the features discussed in this chapter are essential for our purpose of modeling the identified morality facets, it is not our strategy to prepare and develop one complex system with all features discussed. Instead, we may benefit from existing systems with some particular combined features required for modeling only some specific aspects of morality facets we discussed in this chapter.

Apart from counterfactuals and the use of tabling technique in abduction and updating, other components have initially been featured in two existing systems developed earlier, viz., ACORDA (Lopes, 2006; Lopes and Pereira, 2006; Pereira and Lopes, 2009) and its subsequent development, PROBABILISTIC EPA (Han, 2009; Han et al., 2008; Pereira and Han, 2009). These two systems include three main features: abduction, updating, and preferences. PROBABILISTIC EPA is additionally equipped with probabilistic reasoning feature based on probabilistic LP language P-log (Baral et al., 2009). They will be discussed in Chapter 7.

We start, in the subsequent chapter, to first focus on tabling, by looking into the possibility of exploiting its mechanisms for abduction and updating, individually.

CHAPTER 5

TABLING IN ABDUCTION AND UPDATING

In this thesis, we are addressing the interplay amongst appropriate LP features to represent moral facets and to reason about them. One such interplay is between LP abduction and updating, both supported with tabling mechanisms. In this chapter, we propose novel approaches for employing tabling in abduction and updating – *separately* – viz., tabling abductive solutions in contextual abduction (Section 5.1), plus the incremental tabling of fluents for LP updating (Section 5.2). Moreover, these two individual approaches form the components for their subsequent joint tabling technique when combining LP abduction and updating, to be discussed in detail in Chapter 7.

The new techniques introduced here, as well as their later joint combination, are general for normal logic programs, and not specific to morality applications. That is, they are of interest in themselves and may be applicable to other domains.

5.1 Tabling Abductive Solutions in Contextual Abduction

In abduction, it is often the case that abductive solutions found within one context are also relevant in a different context, and can be reused with little cost. As discussed in the previous chapter, the issue of reusing solutions to a goal is commonly addressed in Logic Programming – absent of abduction – by employing a tabling technique (Swift, 1999). Tabling appears to be conceptually suitable for abduction too; in this case, to reuse priorly obtained abductive solutions. In practice, an abductive solution to a goal G is not immediately amenable to tabling, because such a solution is typically attached to G's abductive context. By an *abductive context of a goal G*, we mean a set of abducibles that provides the context in which an abductive solution for G is sought.[1]

[1]The set of abducibles in an abductive context is represented in the sequel using the usual Prolog list notation.

In this section, we discuss a technique for reusing priorly obtained abductive solutions, from one abductive context to another, by benefiting from LP tabling. This technique of *tabling abductive solutions in contextual abduction*, called TABDUAL, is underpinned by ABDUAL (Alferes et al., 2004a), an approach for computing abduction over the Well-Founded Semantics. TABDUAL consists of a program transformation that concretely realizes the abstract theory of ABDUAL, but now in the presence of tabling abductive solutions. It specifically employs the dual program transformation, introduced in ABDUAL, to more efficiently handle the problem of abduction under negative goals.

We start by giving the motivation for the need of tabled abduction, and subsequently show how tabled abduction is conceptualized and realized in the TABDUAL transformation.

Example 3 (Motivation and Idea) *Consider the program P_1 below in an abductive framework $\langle P_1, \{a/0, b/0\}, \emptyset\rangle$:*

$$q \leftarrow a.$$
$$s \leftarrow q, b.$$
$$t \leftarrow s, q.$$

Suppose three queries are invoked, asking for individual explanations of q, s, and t, in that order.

- *The first query, q, is satisfied simply by taking $[a]$ as the abductive solution for q, and tabling it.*
- *Executing the second query, s, amounts to satisfying the two subgoals in its body, i.e., invoking q followed by abducing b. Since q has previously been invoked, we can benefit from reusing its solution, instead of recomputing, given that the solution was tabled. That is, query s can be solved by extending the current ongoing abductive context $[b]$ of subgoal q with the already tabled abductive solution $[a]$ of q, yielding the abductive solution $[a, b]$.*
- *The final query t can be solved similarly. Invoking the first subgoal s results in the priorly registered abductive solution $[a, b]$, which becomes the current abductive context of the second subgoal q. Since $[a, b]$ subsumes the previously obtained (and tabled) abductive solution $[a]$ of q, we can then safely take $[a, b]$ as the abductive solution to query t.*

This example shows how $[a]$, as the abductive solution of the first query q, can be reused from one abductive context of q (viz., $[b]$ in the second query, s) to another context (viz., $[a, b]$ in the third query, t). One may observe that if the body of rule q contains a huge number of subgoals, it may potentially cause an expensive recomputation of its abductive solutions, if they have not been tabled.

TABDUAL comprises two stages. The first stage is a *program transformation* that provides a self-sufficient program transform on which the second stage, the *abduction* itself, is directly enacted through queries.

5.1.1 TABDUAL Program Transformation

Example 3 indicates two key ingredients of the transformation:

- *Abductive contexts*, which relay the ongoing abductive solution from one subgoal to subsequent subgoals in the body of a rule, as well as from the head to the body of a rule, via *input* and *output* contexts.
- *Tabled predicates*, which table the abductive solutions for predicates that appear in the head of rules in the program, such that they can be reused from one abductive context to another.

The TABDUAL program transformation consists of several parts, viz., the transformations for tabling abductive solutions (Section 5.1.1.1), for producing dualized negation (Section 5.1.1.2), and for inserting abducibles into an abductive context (Section 5.1.1.3). This program transformation also requires a given query to be transformed. This is detailed in Section 5.1.1.4.

5.1.1.1 Tabling Abductive Solutions

We continue in Example 4 to show how to realize the idea described in Example 3 through a program transformation. It illustrates how every rule in P_1 is transformed, by introducing a corresponding tabled predicate with one extra argument for an abductive solution entry. The newly introduced tabled predicate therefore essentially tables this abductive solution.

Example 4 *We show first how the rule $t \leftarrow s, q$ in P_1 is transformed into two rules:*

$$t_{ab}(E_2) \leftarrow s([\,], E_1), q(E_1, E_2). \tag{5.1}$$

$$t(I, O) \leftarrow t_{ab}(E), produce\_context(O, I, E). \tag{5.2}$$

Predicate $t_{ab}(E)$ is the tabled predicate which is introduced to table one abductive solution for t in its argument E. Its definition, in the rule on the left, follows from the original definition of t. Two extra arguments, that serve as input and output contexts, are added to the subgoals s and q in the rule's body.

- *Rule 5.1 expresses that the tabled abductive solution E_2 of t_{ab} is obtained by relaying the ongoing abductive solution stored in context E_1 from subgoal s to subgoal q in the body, given the empty input abductive context $[\,]$ of s. This input abductive context is empty because there is no abducible by itself in the body of the original rule of t.*
- *Rule 5.2 shows how the tabled abductive solution in E of t_{ab} can be reused for a given (input) abductive context of t. This rule expresses that the output abductive solution O of t is obtained from the solution entry E of t_{ab} and the given input context I of t, via the* TABDUAL *system predicate $produce\_context(O, I, E)$.*

The system predicate produce_context(O, I, E) should guarantee that it produces a consistent *output O from the input abductive context I and the abductive solution entry E, encompassing both.*

The other two rules in P_1 are transformed following the same idea. The rule $s \leftarrow q, b$ is transformed into:

$$s_{ab}(E) \leftarrow q([b], E). \tag{5.3}$$
$$s(I, O) \leftarrow s_{ab}(E), produce\_context(O, I, E). \tag{5.4}$$

where $s_{ab}(E)$ is the predicate that tables, in E, the abductive solution of s. Notice how b, the abducible appearing in the body of the original rule of s, becomes the input abductive context of q. The same transformation is obtained, even if b comes before q in the body of the rule s.

Finally, the rule $q \leftarrow a$ is transformed into:

$$q_{ab}([a]).$$
$$q(I, O) \leftarrow q_{ab}(E), produce\_context(O, I, E).$$

where the original rule of q, which is defined solely by the abducible a, is simply transformed into the tabled fact $q_{ab}/1$.

The transformation for tabling abductive solutions is formalized in Definition 17.

In the sequel, we write $\bar{t}$ to denote $[t_1, \ldots, t_n]$, $n \geq 0$. For a predicate p/n, we write $p(\bar{t})$ to denote $p(t_1, \ldots, t_n)$. In particular, we write $\bar{X}$ to denote $[X_1, \ldots, X_n]$, $p(\bar{X})$ to denote $p(X_1, \ldots, X_n)$, and $p(\bar{X}, Y, Z)$ to denote $p(X_1, \ldots, X_n, Y, Z)$, where all variables are distinct.

Definition 17 (Transformation: tabling abductive solutions) *Consider an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$. By H_r and $\mathcal{B}_r$, we refer to the head and the body of rule $r \in P$, respectively. Let $\mathcal{A}_r \subseteq \mathcal{B}_r$ be the set of abducibles (either positive or negative) in $r \in P$, and r' be the rule, such that $H_{r'} = H_r$ and $\mathcal{B}_{r'} = \mathcal{B}_r \setminus \mathcal{A}_r$.*

1. *For every rule $r \in P$ with r' the rule $l(\bar{t}) \leftarrow L_1, \ldots, L_m$, we define $\tau'(r)$:*

$$l_{ab}(\bar{t}, E_m) \leftarrow \alpha(L_1), \ldots, \alpha(L_m).$$

where α is defined as:

$$\alpha(L_i) = \begin{cases} l_i(\bar{t}_i, E_{i-1}, E_i) & \text{, if } L_i \text{ is an atom } L_i = l_i(\bar{t}_i) \\ not\_l_i(\bar{t}_i, E_{i-1}, E_i) & \text{, if } L_i \text{ is a default literal } L_i = not\ l_i(\bar{t}_i) \end{cases}$$

with $1 \leq i \leq m$, E_i are fresh variables and $E_0 = \mathcal{A}_r$. Note that variables E_is serve as abductive contexts*.*

2. *For every predicate p/n with rules in P (i.e., rules whose heads is the predicate p/n), we define $\tau^+(p)$:*

$$p(\bar{X}, I, O) \leftarrow p_{ab}(\bar{X}, E),\ produce\_context(O, I, E).$$

where $produce\_context(O, I, E)$ is a TABDUAL *system predicate that concerns itself with: whether E is already contained in I and, if not, whether there are any abducibles from E, consistent with I, that can be added to produce O. If E is inconsistent with I then the specific entry E cannot be reused with I, $produce\_context/3$ fails and another entry E is sought.*

Example 5 *Consider an abductive framework $\langle P, \{a/1\}, \emptyset \rangle$, where the program P (whose rules are named with r_i) is given below:*

$$\begin{array}{ll} r_1: & u(0,\_). \\ r_2: & u(s(X),Y) \leftarrow a(X), v(X,Y,Z), not\ w(Z). \\ r_3: & v(X,X,s(X)). \end{array}$$

We have $\mathcal{A}_{r_i}$ and r_i', for $1 \leq i \leq 3$, as follows:

- $\mathcal{A}_{r_1} = [\,]$ *and* $r_1': \ u(0,\_).$
- $\mathcal{A}_{r_2} = [a(X)]$ *and* $r_2': \ u(s(X),Y) \leftarrow v(X,Y,Z), not\ w(Z).$
- $\mathcal{A}_{r_3} = [\,]$ *and* $r_3': \ v(X,X,s(X)).$

The transformation of Definition 17 results in:

$$\begin{array}{lll} \tau'(r_1): & u_{ab}(0,\_,[\,]). & \\ \tau'(r_2): & u_{ab}(s(X),Y,E_2) & \leftarrow \ v(X,Y,Z,[a(X)],E_1),\ not\_w(Z,E_1,E_2). \\ \tau'(r_3): & v_{ab}(X,X,s(X),[\,]). & \end{array}$$

$$\begin{array}{lll} \tau^+(u): & u(X_1,X_2,I,O) & \leftarrow \ u_{ab}(X_1,X_2,E),\ produce\_context(O,I,E). \\ \tau^+(v): & v(X_1,X_2,X_3,I,O) & \leftarrow \ v_{ab}(X_1,X_2,X_3,E),\ produce\_context(O,I,E). \end{array}$$

Observe that both arguments of $u/2$ are kept in the tabled predicate u_{ab} (as its first two arguments), and one extra argument is added (as its third argument) for tabling its abductive solution entry. Similar transformation also applies to $v/3$. The transformation does not create (and indeed do not need) $\tau^+(w)$, because there is no rule whose head is $w/1$ in the program P.

5.1.1.2 Abduction under Negative Goals

For abduction under negative goals, the program transformation employs the *dual program transformation* of ABDUAL (Alferes et al., 2004a). In this transformation, negative goals are syntactically treated as new atoms. The motivation behind the transformation is to enable us to obtain the solutions of a negative goal *not G* without having to compute all abductive solutions of the positive goal G and subsequently to negate their disjunction. In other words, with the dual program transformation, the abductive solutions of a negative goal can be obtained one at a time, as we treat abduction under positive goals.

The idea of the dual program transformation is to define, for each atom A and its (possibly empty) set of rules $\mathcal{R}_A$ in a program P, a set of dual rules whose head is not_A, such that not_A is true if and only if A is false by $\mathcal{R}_A$ in the employed semantics of P.

Note that, instead of having a negative goal *not A* as the head of a dual rule, we use its corresponding atom *not_A*. Example 6 illustrates this main idea of the dual transformation is realized in TABDUAL by means of two-layer dual rules.

Example 6 *Consider an abductive framework $\langle P_2, \{a/0\}, \emptyset\rangle$, where the program P_2 is as follows:*

$$p \leftarrow a. \tag{5.5}$$

$$p \leftarrow q, not\ r. \tag{5.6}$$

$$r.$$

- *With regard to atom p, the transformation creates a set of dual rules for p which falsify p with respect to its two rules. That is, both rules 5.5* and *5.6 are falsified, as expressed below by predicate p^{*1} and p^{*2}, respectively:*

$$not\_p(T_0, T_2) \leftarrow p^{*1}(T_0, T_1),\ p^{*2}(T_1, T_2).$$

  We refer this resulting single rule as the first layer *of the dual program transformation (or, the first layer dual rule). In this rule, input and output abductive context arguments, T_0 and T_2, respectively, are added in the head. Similarly, these context arguments are added into each subgoal of the rule's body, where intermediate context T_1 relays the ongoing abductive solution from p^{*1} to p^{*2}.*

  The second layer *contains the definitions of p^{*1} and p^{*2}, where p^{*1} and p^{*2} are defined by falsifying the body of p's first rule (5.5) and second one (5.6), respectively.*

  - *In case of p^{*1}, rule 5.5 is falsified only by abducing the negation of the abducible a. Therefore, we have:*

$$p^{*1}(I, O) \leftarrow a^*(I, O).$$

    Notice that the negation a^* of the abducible a refers to its own abduction, which is achieved by invoking the subgoal $a^*(I, O)$. This subgoal is defined via the transformation of abducibles, which will be discussed in Section 5.1.1.3.

  - *In case of p^{*2}, rule 5.6 is falsified by alternatively failing one subgoal in its body at a time, viz., by negating q or by negating not r:*

$$p^{*2}(I, O) \leftarrow not\_q(I, O). \qquad p^{*2}(I, O) \leftarrow r(I, O).$$

- *With regard to atom q, the dual program transformation produces the fact:*

$$not\_q(I, I).$$

  as its dual rule for an obvious reason, that there is no rule with the head q in P_2. As a fact, the content of the context I in this dual rule is simply relayed from the input to the output abductive context. That is, having an empty body, the output abductive context does not depend on the context of any other goals, but depends only on its corresponding input abductive context.

- *With regard to atom r, since r is a fact, in principle it produces the first layer rule:*

$$not\_r(T_0, T_1) \leftarrow r^{*1}(T_0, T_1).$$

but with no definition of $r^{*1}/2$. In other words, invoking $not\_r(\_, \_)$ will vacuously fail.

This example particularly shows how the dual rules for nullary predicates are derived, viz., by falsifying the bodies of their corresponding positive rules. In the case of non-nullary predicates, a goal may also fail (or equivalently, its negation succeeds), when its arguments disagree with the arguments of its rules. For instance, if we have just a fact $q(1)$, then goal $q(0)$ will fail (or equivalently, goal *not* $q(0)$ succeeds). It therefore provides some guidance on how to treat non-nullary predicates in the dual program transformation. That is, besides falsifying the body of a rule, a dual of a non-nullary predicate can additionally be defined by disunifying its arguments and the arguments of its corresponding source rule, as illustrated in Example 7.

Example 7 *Consider an abductive framework $\langle P_3, \{a/1\}, \emptyset \rangle$, where the program P_3 is as follows:*

$$q(0). \tag{5.7}$$
$$q(s(X)) \leftarrow a(X). \tag{5.8}$$

The dual program transformation of non-nullary predicate $q/1$ is given below:

$$\begin{array}{lll} 1. & not\_q(X, T_0, T_2) & \leftarrow \quad q^{*1}(X, T_0, T_1), q^{*2}(X, T_1, T_2). \\ 2. & q^{*1}(X, I, I) & \leftarrow \quad X \setminus = 0. \\ 3. & q^{*2}(X, I, I) & \leftarrow \quad X \setminus = s(\_). \\ 4. & q^{*2}(s(X), I, O) & \leftarrow \quad a^*(X, I, O). \end{array}$$

Line 1 shows the first layer dual rule for predicate $q/1$, which is defined as usual, i.e., $q/1$ is falsified by falsifying both rules 5.7 and 5.8. Lines 2-4 show the second layer dual rules for $q/1$:

- In case of q^{*1}, the first rule of $q/1$, which is fact $q(0)$, is falsified by disunifying q^{*1}'s argument X with 0 (line 2). This is the only way to falsify rule 5.7, since it has no body.

- In case of q^{*2}, the second rule of $q/1$ is falsified by disunifying q^{*2}'s argument X with the term $s(\_)$ (line 3). Or, it can alternatively be falsified by keeping the head's argument, but falsifying its body, i.e., by abducing the negation of the abducible $a/1$ (line 4).

We next specify, in Definition 18, the transformation that constructs the two-layer dual rules in TABDUAL.

Definition 18 (Transformation: constructing dual rules) *Consider an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$. Let P^+ be $P \cup \mathcal{IC}$.*

1. *For every predicate p/n, $n \geq 0$, whose rules in P^+ are as follows:*

$$\begin{array}{lcl} p(\bar{t}_1) & \leftarrow & L_{11}, \ldots, L_{1n_1}. \\ & \vdots & \\ p(\bar{t}_m) & \leftarrow & L_{m1}, \ldots, L_{mn_m}. \end{array}$$

with $n_i \geq 0$, $1 \leq i \leq m$:

   a) *The first layer dual rule is defined by $\tau^-(p)$:*

$$not\_p(\bar{X}, T_0, T_m) \leftarrow p^{*1}(\bar{X}, T_0, T_1), \ldots, p^{*m}(\bar{X}, T_{m-1}, T_m).$$

   with T_i, $0 \leq i \leq m$, are fresh variables. Note that variables T_is serve as abductive contexts.

   b) *The second layer dual rules are defined by:*
   $\tau^*(p) = \bigcup_{i=1}^{m} \tau^{*i}(p)$ and $\tau^{*i}(p)$ is the smallest set that contains the following rules:

$$\begin{array}{lcl} p^{*i}(\bar{X}, I, I) & \leftarrow & \bar{X} \neq \bar{t}_i. \\ p^{*i}(\bar{t}_i, I, O) & \leftarrow & \sigma(L_{i1}, I, O). \\ & \vdots & \\ p^{*i}(\bar{t}_i, I, O) & \leftarrow & \sigma(L_{in_i}, I, O). \end{array}$$

   where σ is defined as follows:

$$\sigma(L_{ij}, I, O) = \begin{cases} l_{ij}(\bar{t}_{ij}, I, O) & \text{, if } L_i \text{ is a default literal not } l_{ij}(\bar{t}_{ij}) \text{ or} \\ & \quad \text{a negative abducible } {l_{ij}}^*(\bar{t}_{ij}) \\ not\_l_{ij}(\bar{t}_{ij}, I, O) & \text{, if } L_i \text{ is an atom } l_{ij}(\bar{t}_{ij}) \\ {l_{ij}}^*(\bar{t}_{ij}, I, O) & \text{, if } L_i \text{ is a positive abducible } l_{ij}(\bar{t}_{ij}) \end{cases}$$

Notice that, in case of $p/0$ (i.e. $n = 0$), rule $p^{*i}(\bar{X}, I, I) \leftarrow \bar{X} \neq \bar{t}_i$ is omitted, since both $\bar{X}$ and $\bar{t}_i$ are $[\,]$. This means, when $p/0$ is defined as a fact in P^+, we have $not\_p(T_0, T_1) \leftarrow p^{*1}(T_0, T_1)$ in the first layer, but there is no rule of $p^{*1}/2$ in the second layer (cf. the dual rule of predicate $r/0$ in Example 6).

2. *For every predicate r/n in P^+ ($n \geq 0$) that has no rule, we define $\tau^-(r)$:*

$$not\_r(\bar{X}, I, I).$$

In particular, if $\mathcal{IC} = \emptyset$, we have $\tau^-(\bot) : not\_\bot(I, I)$.

Example 8 *Recall Example 5. The transformation of Definition 18 results in:*

$$\begin{array}{rl}
\tau^{-}(u): & not_u(X_1, X_2, T_0, T_2) \leftarrow u^{*1}(X_1, X_2, T_0, T_1),\ u^{*2}(X_1, X_2, T_1, T_2).\\
\tau^{-}(v): & not_v(X_1, X_2, X_3, T_0, T_1) \leftarrow v^{*1}(X_1, X_2, X_3, T_0, T_1).\\
\tau^{-}(w): & not_w(X, I, I).\\
\tau^{-}(\bot): & not_\bot(X, I, I).
\end{array}$$

$$\begin{array}{rl}
\tau^{*}(u): & u^{*1}(X_1, X_2, I, I) \leftarrow [X_1, X_2] \neq [0, _].\\
& u^{*2}(X_1, X_2, I, I) \leftarrow [X_1, X_2] \neq [s(X), Y].\\
& u^{*2}(s(X), Y, I, O) \leftarrow a^{*}(X, I, O).\\
& u^{*2}(s(X), Y, I, O) \leftarrow not_v(X, Y, Z, I, O).\\
& u^{*2}(s(X), Y, I, O) \leftarrow w(Z, I, O).\\
\tau^{*}(v): & v^{*1}(X_1, X_2, X_3, I, I) \leftarrow [X_1, X_2, X_3] \neq [X, X, s(X)].
\end{array}$$

5.1.1.3 Transforming Abducibles

In Example 6, $p^{*1}(I, O)$ is defined by abducing a^{*}, achieved by invoking subgoal $a^{*}(I, O)$. Abduction in TABDUAL is realized by transforming each abducible into a rule that updates the abductive context with the transformed abducible. For instance, abducible a of Example 6 translates to:

$$a(I, O) \leftarrow insert\_abducible(a, I, O).$$

where *insert_abducible*/3 is a TABDUAL system predicate that inserts the given abducible into the abductive context, described in Definition 19 below. Abducible a^{*} is transformed similarly.

The specification for the transformation of abducibles is given in Definition 19.

Definition 19 (Transformation of abducibles) *Given an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC}\rangle$. For every $a/n \in \mathcal{AB}$, we define $\tau^{\circ}(a)$ as the smallest set that contains the rules:*

$$\begin{array}{lcl}
a(X_1, \ldots, X_n, I, O) & \leftarrow & insert_abducible(a(X_1, \ldots, X_n), I, O).\\
a^{*}(X_1, \ldots, X_n, I, O) & \leftarrow & insert_abducible(a^{*}(X_1, \ldots, X_n), I, O).
\end{array}$$

where $insert\_abducible(A, I, O)$ is a TABDUAL *system predicate a* TABDUAL *system that inserts the abducible A into the input context I, resulting in the output context O. It maintains the consistency of the abductive context, failing if inserting A results in an inconsistent one.*

Example 9 *Recall Example 5. The transformation of Definition 19 results in:*

$$\begin{array}{rlcl}
\tau^{\circ}(a(X)): & a(X, I, O) & \leftarrow & insert_abducible(a(X), I, O).\\
& a^{*}(X, I, O) & \leftarrow & insert_abducible(a^{*}(X), I, O).
\end{array}$$

The specification of the complete TABDUAL program transformation is given in Definition 20.

Definition 20 (TABDUAL transformation) *Let $\mathcal{F}$=$\langle P, \mathcal{AB}, \mathcal{IC}\rangle$ be an abductive framework, $\mathcal{P}$ be the set of predicates in P and $P^{+} = P \cup \mathcal{IC}$. Taking:*

- $\tau'(\mathcal{F}) = \{\tau'(r) \mid r \in P\}$
- $\tau^+(\mathcal{F}) = \{\tau^+(p) \mid p \in \mathcal{P}$ *with rules in* $P\}$
- $\tau^-(\mathcal{F}) = \{\tau^-(p) \mid p \in \mathcal{P} \cup \{\bot\}\}$
- $\tau^*(\mathcal{F}) = \{\tau^*(p) \mid p \in \mathcal{P} \cup \{\bot\}$ *with rules in* $P^+\}$
- $\tau^\circ(\mathcal{F}) = \{\tau^\circ(a) \mid a \in \mathcal{AB}\}$

The TABDUAL *transformation* $\tau(\mathcal{F})$ *is defined as:*

$$\tau(\mathcal{F}) = \tau'(\mathcal{F}) \cup \tau^+(\mathcal{F}) \cup \tau^-(\mathcal{F}) \cup \tau^*(\mathcal{F}) \cup \tau^\circ(\mathcal{F})$$

Example 10 *The set of rules obtained in Example 5, 8, and 9 forms* $\tau(\mathcal{F})$ *of the abductive framework* $\mathcal{F}$=$\langle P, \mathcal{AB}, \mathcal{IC}\rangle$.

5.1.1.4 Transforming queries

A query to a program, consequently, should be transformed:

- A positive goal G is simply augmented with the two extra arguments for the input and output abductive contexts.
- A negative goal *not* G is renamed into not_G, and added the two extra (input and output) abductive context arguments.

Moreover, a query should additionally ensure that all integrity constraints are satisfied. Note that when there is no integrity constraint, then, following Definition 18, the following fact is added into the transform:

$$not\_\bot(I, I).$$

Otherwise, integrity constraints are transformed just like any other rules, omitting the transformed rules with the heads $\bot_{ab}(E)$ and $\bot(I, O)$.

Finally, a query should always be conjoined with $not\_\bot/2$ to ensure that all integrity constraints are satisfied.

Example 11 *Query*

$$?-\ not\ p.$$

first transforms into $not\_p(I, O)$. *Then, to satisfy all integrity constraints, it is conjoined with* $not\_\bot/2$, *resulting in top goal:*

$$?-\ not\_p([\,], T),\ not\_\bot(T, O).$$

where O is an abductive solution to the query, given initially an empty input abductive context. Note, how O is obtained by further constraining the output abductive context T for not_p, *via passing it to the subsequent subgoal* $not\_\bot$ *for confirmation.*

Definition 21 provides the specification of the query transformation.

Definition 21 (Transformation of queries) *Let $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ be an abductive framework and Q be a query:*

$$?-\ G_1,\ \ldots,\ G_m.$$

TABDUAL *transforms query Q into $\Delta(Q)$:*

$$?-\ \delta(G_1),\ \ldots,\ \delta(G_m), not\_\bot(T_m, O).$$

where δ is defined as:

$$\delta(G_i) = \begin{cases} g_i(\bar{t}_i, T_{i-1}, T_i) & , \text{if } G_i = g_i(\bar{t}_i) \\ not\_g_i(\bar{t}_i, T_{i-1}, T_i) & , \text{if } G_i = not\ g_i(\bar{t}_i) \end{cases}$$

T_0 is a given initial abductive context (or an empty context $[\,]$, by default), $1 \leq i \leq m$, T_i, O are fresh variables. The output abductive context O returns the abductive solution(s) of the query.

Example 12 *Recall Example 5. Query:*

$$?-\ u(0, s(0)),\ not\ u(s(0), 0).$$

is transformed by Definition 21 into:

$$?-\ u(0, s(0), [\,], T_1),\ not\_u(s(0), 0, T_1, T_2),\ not\_\bot(T_2, O).$$

5.1.2 Implementation Aspects

TABDUAL is implemented in XSB Prolog (Swift and Warren, 2012), and many of its implementation aspects benefit from features of XSB Prolog.

5.1.2.1 Grounding Dualized Negated Subgoals

Example 13 *Consider an abductive framework $\langle P_4, \{a/1\}, \mathcal{IC}_4 \rangle$, where the program P_4 is as follows:*

$$q(1).$$
$$r(X) \leftarrow a(X).$$

and $\mathcal{IC}_4$:

$$\bot \leftarrow q(X), r(X).$$

The TABDUAL *transformation results in:*

1. $q_{ab}(1,[\,]).$
2. $q(X,I,O) \leftarrow q_{ab}(X,E), produce\_context(O,I,E).$
3. $not\_q(X,I,O) \leftarrow q^{*1}(X,I,O).$
4. $q^{*1}(X,I,I) \leftarrow X \setminus = 1.$

5. $r_{ab}(X,[a(X)]).$
6. $r(X,I,O) \leftarrow r_{ab}(X,E), produce\_context(O,I,E).$
7. $not\_r(X,I,O) \leftarrow r^{*1}(X,I,O).$
8. $r^{*1}(X,I,I) \leftarrow X \setminus = \_.$
9. $r^{*1}(X,I,O) \leftarrow a^*(X,I,O).$

10. $not\_\bot(I,O) \leftarrow \bot^{*1}(I,O).$
11. $\bot^{*1}(I,O) \leftarrow not\_q(X,I,O).$
12. $\bot^{*1}(I,O) \leftarrow not\_r(X,I,O).$

Consider query $q(1)$, which is transformed into:

$$?-\ q(1,[\,],T), not\_\bot(T,O).$$

Invoking the first subgoal, $q(1,[\,],T)$, results in $T = [\,]$. Invoking subsequently the second subgoal, $not\_\bot([\,],O)$, results in the abductive solution of the given query: $O = [a^*(X)]$, obtained via rules 10, 12, 7, and 9. Note that rule 11, an alternative to $\bot^{*1}$, fails due to uninstantiated X in its subgoal $not\_q(X,I,O)$, which leads to failing rules 3 and 4. For the same reason, rule 8, an alternative to r^{*1}, also fails.

Instead of having $[a^*(1)]$ as the abductive solution to the query $q(1)$, we have a non-ground abductive solution $[a^*(X)]$. It does not meet our requirement, in Section 4.2, that abducibles must be ground on the occasion of their abduction. The problem can be remedied by instantiating X, in rule 12, thereby eventually grounding the abducible $a^*(X)$ when it is abduced, i.e., the argument X of subgoal $a^*/3$, in rule 9, becomes instantiated.

In the implementation, grounding a dualized negated subgoal is achieved as follows: in addition to placing a negated literal, say not_p, in the body of the second layer dual rule, all positive literals that precede literal p, in the body of the corresponding source rule, are also kept in the body of the dual rule. For rule 12, introducing the positive subgoal $q(X)$, originating from the source rule, before the negated subgoal $not\_r(X,I,O)$ in the body of rule 12, helps instantiate X in this case. Rule 12 now becomes (all other rules remain the same):

$$\bot^{*1}(I,O) \leftarrow q(X,I,T), not\_r(X,T,O).$$

Notice that, differently from before, the rule is now defined by introducing all positive literals that appear before r in the original rule; in this case we introduce $q/3$ before $not\_r/3$. As the result, the argument X in $not\_r/3$ is instantiated to 1, due to the invocation of $q/3$,

just like the case in the source rule. It eventually helps ground the the negated abducible $a^*(X)$, when it is abduced, and the correct abductive solution $[a^*(1)]$ to query $q(1)$ is returned. By implementing this technique, we are also able to deal with non-ground positive goals, e.g., query $q(X)$ gives the correct abductive solution as well, i.e. $[a^*(1)]$ for $X = 1$.

There are some points to remark on regarding this implementation technique:

- The semantics of dual rules does not change because the conditions for failure of their source counterpart rules are that one literal must fail, even if the others succeed. The cases where the others do not succeed are handled in the other alternatives of dual rules.
- This technique may benefit from the TABDUAL's tabled predicate, e.g. q_{ab} for predicate q, as it helps avoid redundant derivations of the newly introduced positive literals in dual rules.
- Information about shared variables in the body and whether they are local or not, may be useful to avoid introducing positive literals that are not contributing to further grounding.

5.1.2.2 Dealing with Non-Ground Negative Goals

Example 14 *Consider an abductive framework $\langle P_5, \{a/1\}, \emptyset\rangle$, where the program P_5 is as follows:*

$$p(1) \leftarrow a(1).$$
$$p(2) \leftarrow a(2).$$

Query $p(X)$ succeeds under TABDUAL, giving two abductive solutions, viz., $[a(1)]$ and $[a(2)]$ for $X = 1$ and $X = 2$, respectively. But query *not* $p(X)$ does not deliver the expected solution. Instead of returning the abductive solution $[a^*(1), a^*(2)]$ for any instantiation of X, it returns $[a^*(1)]$ for a particular $X = 1$. In order to find the problem, we first look into the definition of $not\_p/3$:

1. $not\_p(X, I, O) \leftarrow p^{*1}(X, I, T), p^{*2}(X, T, O).$
2. $p^{*1}(X, I, I) \leftarrow X \setminus= 1.$
3. $p^{*1}(1, I, O) \leftarrow a^*(1, I, O).$
4. $p^{*2}(X, I, I) \leftarrow X \setminus= 2.$
5. $p^{*2}(2, I, O) \leftarrow a^*(2, I, O).$

Recall that query $?-\ not\ p(X)$ is transformed into:

$$?-\ not\_p(X, [\,], N), not\_\bot(N, O).$$

When the goal $not\_p(X, [\,], N)$ is launched, it first invokes $p^{*1}(X, [\,], T)$. It succeeds by the second rule of p^{*1}, in line 3 (the first rule, in line 2, fails it), with variable X is instantiated

to 1 and T to $[a^*(1)]$. The second subgoal of $not\_p(X, [\,], N)$ is subsequently invoked with the same instantiation of X and T, i.e. $p^{*2}(1, [a^*(1)], O)$, and it succeeds by the first rule of p^{*2}, in line 4, and results in $N = [a^*(1)]$. Since there is no integrity constraint, i.e., $\mathcal{IC}_5 = \emptyset$, the abductive solution $[a^*(1)]$ is just relayed from N to O, due to the dual $not\_\bot(I, I)$ in the transformed program (see Definition 18), thus returning the abductive solution $[a^*(1)]$ for the original query `?-` $not\ p(X)$, where X is instantiated to 1.

The culprit is that both subgoals of $not\_p/3$, viz., $p^{*1}/3$ and $p^{*2}/3$, share the argument X of $p/1$. This should not be the case, as $p^{*1}/3$ and $p^{*2}/3$ are derived from two different rules of $p/1$, hence failing p should be achieved by invoking p^{*1} and p^{*2} with an independent argument X. In other words, different variants of the calling argument X should be used in $p^{*1}/3$ and $p^{*2}/3$, as shown for rule $not\_p/3$ (line 1) below:

$$\begin{aligned} not\_p(X, T_0, T_2) \quad \leftarrow \quad & copy\_term([X], [X_1]), p^{*1}(X_1, T_0, T_1), \\ & copy\_term([X], [X_2]), p^{*2}(X_2, T_1, T_2). \end{aligned}$$

where the Prolog built-in predicate $copy\_term/2$ provides a variant of the list of arguments; in this example, we simply have only one argument, i.e. $[X]$.

Now, $p^{*1}/3$ and $p^{*2}/3$ are invoked using variant independent calling arguments, viz., X_1 and X_2, respectively. The same query first invokes $p^{*1}(X_1, [\,], T_1)$, which results in $X_1 = 1$ and $T_1 = [a^*(1)]$ (by rule 3), and subsequently invokes $p^{*2}(X_2, [a^*(1)], T_2)$, resulting in $X_2 = 2$ and $T_2 = [a^*(1), a^*(2)]$ (by rule 5). It eventually ends up with the expected abductive solution: $[not\ a(1), not\ a(2)]$ for any instantiation of X, i.e., X remains unbound.

The technique ensures, as this example shows, that query `?-` $p(X)$ fails for every X, and its negation, `?-` $not\ p(X)$, hence succeeds. The dual rules produced for the negation are tailored to be, by definition, an 'if and only if' with regard to their corresponding source rules. If we added the fact $p(Y)$ to P_5, then the same query `?-` $not\ p(X)$ would not succeed because now we have the first layer dual rule:

$$\begin{aligned} not\_p(X, T_0, T_3) \quad \leftarrow \quad & copy\_term([X], [X_1]), p^{*1}(X_1, T_0, T_1), \\ & copy\_term([X], [X_2]), p^{*2}(X_2, T_1, T_2), \\ & copy\_term([X], [X_3]), p^{*3}(X_3, T_2, T_3). \end{aligned}$$

and an additional second layer dual rule $p^{*3}(X, \_, \_) \leftarrow X \neq \_$ that always fails; its abductive contexts are thus irrelevant.

5.1.2.3 Transforming Predicates Comprising Just Facts

TABDUAL transforms predicates that comprise just facts as any other rules in the program. For instance, see fact $q(1)$ and its transformed rules (rules 1-4), in Example 13. This is clearly superfluous as facts do not induce any abduction. For programs with large factual data, a simpler transformation for predicates that consist of just facts is desirable.

Suppose a predicate $q/1$ consists of just facts:

$$q(1). \qquad q(2). \qquad q(3).$$

Rather than applying the transformation defined in Definition 17, rules $q_{ab}/2$ and $q/3$ can be substituted by a single rule:

$$q(X, I, I) \leftarrow q(X).$$

and their negations, rather than using dual rules as obtained by Definition 18, can be defined to a single rule:

$$not\_q(X, I, I) \leftarrow not\ q(X).$$

Note that the input and output context arguments are added in the head, and the input context is just passed intact to the output context. Both rules simply execute the fact calls.

Facts of predicate $q/1$ can thus be defined in the so-called *non-abductive part* of the source program, independently of the number of facts $q/1$ are there in the program. The non-abductive part is distinguished from the abductive part by the beginProlog and endProlog identifiers. Any program between these identifiers will not be transformed, i.e., it is treated as a usual Prolog program. For the above facts of $q/1$, they are listed in the non-abductive part as:

beginProlog $q(1).$ $q(2).$ $q(3).$ endProlog

5.1.2.4 Dual Transformation by Need

The TABDUAL transformation conceptually constructs *all* (first and second layer) dual rules, in advance, for every defined atom in an input program, regardless whether they are needed in abduction. We refer to this conceptual construction of dual rules in the sequel as the STANDARD dual program transformation.

The STANDARD dual program transformation should be avoided in practice, as potentially large sets of dual rules are created in the transformation, though only a few of them might be invoked during abduction. Consequently, it may burden the dual program transformation itself, affecting the time required to produce dual rules and the space required for the large thus produced transformed.

One solution to this problem is to compute dual rules only *by need*. That is, dual rules are concretely created in the abduction stage (rather than in the transformation stage), based on the need of the on-going invoked goals. The transformed program still contains the single first layer dual rule, but its second layer is defined using a newly introduced TABDUAL system predicate, which will be interpreted by the TABDUAL system on-the-fly, during the abduction stage, to produce the concrete rule definitions of the second layer.

Example 15 *Recall Example 6. The dual transformation by need contains the same first layer:* $not\_p(T_0, T_2) \leftarrow p^{*1}(T_0, T_1), p^{*2}(T_1, T_2)$. *But the second now contains, for each* $i \in \{1, 2\}$*:*

$$p^{*i}(I, O) \leftarrow dual(i, p, I, O). \tag{5.9}$$

Predicate $dual/4$ is a TABDUAL system predicate, introduced to facilitate the dual transformation by need:

1. It constructs *generic* dual rules, i.e., dual rules without specific context assigned to them, by need, from the i-th rule of $p/0$. This construction is performed during abduction.

2. It instantiates the generic dual rules with the provided arguments and input context.

3. Finally, it subsequently invokes the instantiated dual rules.

The concrete definition of this predicate $dual/4$ depends on the dual rules construction mechanism detailed below.

While the dual program transformation by need minimizes the number of the second layer dual rules, constructing dual rules on-the-fly introduces some extra cost in the abduction stage. Such extra cost can be reduced by memoizing the already constructed generic dual rules. Therefore, when such dual rules are later needed, they are available for reuse and their recomputation avoided.

We examine two approaches for memoizing generic dual rules for the dual transformation by need. They lead to different definitions of the system predicate $dual/4$, particularly concerning how generic dual rules are constructed by need. The first approach benefits from tabling to memoize generic dual rules, whereas the second one employs the XSB Prolog's trie data structure (Swift et al., 2015). They are referred in the sequel as BY-NEED(EAGER) and BY-NEED(LAZY), respectively, due to their dual rules construction mechanisms.

Dualization BY-NEED(EAGER): tabling generic dual rules. The straightforward choice for memoizing generic dual rules is to use tabling. The system predicate $dual/4$ in rule 5.9 is defined as follows:

$$dual(N, P, I, O) \leftarrow dual\_rule(N, P, Dual), call\_dual(P, I, O, Dual).$$

where $dual\_rule/3$ is a *tabled* predicate that constructs a generic dual rule $Dual$ from the N-th rule of atom P, and $call\_dual/4$ instantiates $Dual$ with the provided arguments of P and the input context I. It eventually invokes the instantiated dual rule to produce the abductive solution in O.

Though predicate $dual/4$ helps realize the construction of dual rules by need, i.e., only when a particular p^{*i} is invoked, this approach results in an *eager* construction of all dual rules for the i-th rule of predicate p, when local table scheduling, like the one employed by default in XSB Prolog (Swift and Warren, 2012), is in place. This scheduling strategy does not return any answers out of a strongly connected component (SCC) in the subgoal dependency graph, until that SCC is completely evaluated (Swift and Warren, 2012).

As an illustration, in Example 6, when $p^{*2}(I, O)$ is invoked, which subsequently invokes $dual\_rule(2, p, Dual)$, all two alternatives of dual rules from the second rule of p, viz., $p^{*2}(I, O) \leftarrow not\_q(I, O)$ and $p^{*2}(I, O) \leftarrow r(I, O)$ are constructed before $call\_dual/4$ is invoked for each of them. This is a bit against the spirit of a full dual transformation

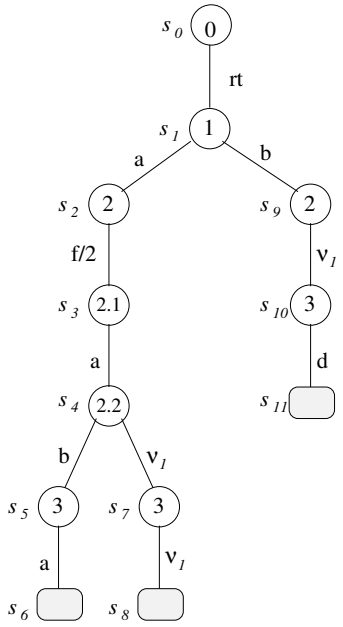


Figure 5.1: Facts stored as a trie

by need, whereby only one alternative dual rule is constructed at a time, just before it is invoked. That is, generic dual rules should rather be constructed *lazily*.

As an alternative to local table scheduling, batched scheduling is also implemented in XSB Prolog, which allows returning answers outside of a maximal SCC as they are derived. In terms of the dual rules construction by need, this means $dual\_rule/3$ would allow dual rules to be lazily constructed. That is, only one generic dual rule is produced at a time before it is instantiated and invoked. Since the choice between the two scheduling strategies can only be made for the whole XSB Prolog installation, and is not (as yet) predicate switchable, we pursue another approach to implement lazy dual rule construction.

Dualization BY-NEED(LAZY): storing generic dual rules in a trie. *Trie* is a tree data structure that allows data, such as strings, to be compactly stored by a shared representation of their prefixes. That is, all the descendants of a node in a trie have a common prefix of the string associated with that node.

XSB Prolog offers a mechanism for facts to be directly stored and manipulated in tries. Figure 5.1, taken from Swift et al. (2015), depicts a trie that stores a set of Prolog facts:

$$\{rt(a, f(a, b), a), rt(a, f(a, X), Y), rt(b, V, d)\}.$$

For trie-dynamic code, trie storage has advantages, both in terms of space and time (Swift et al., 2015):

- A trie can use much less space to store many sets of facts than standard dynamic code, as there is no distinction between the index and the code itself.

- Directly inserting into or deleting from a trie is faster (up to 4-5 times) than with standard dynamic code, as discrimination can be made on a position anywhere in a fact.

XSB Prolog provides predicates for inserting terms into a trie, unifying a term with terms in a trie, and other trie manipulation predicates, both in the low-level and high-level interface.

Generic dual rules can be represented as facts, thus once they are constructed, they can be memoized and later (a copy) retrieved and reused. Given the aforementioned advantages for storing dynamic facts and XSB Prolog's support for its manipulation, a trie is preferable to the common Prolog database to store dynamically generated (i.e., by need) dual rules. The availability of XSB Prolog's system predicates to manipulate terms in a trie permits explicit control in lazily constructing generic dual rules compared to the more eager tabling approach, as detailed below.

A fact of the form $d(N, P, Dual, Pos)$ is used to represent a generic dual rule $Dual$ from the N-th rule of P with the additional tracking information Pos, which informs the position of the literal used in constructing each dual rule. For now, we opt for the low-level trie manipulation predicates, as they can be faster than the higher-level ones.

Using this approach, the system predicate $dual/4$ in rule 5.9 is defined as follows:

1. $dual(N, P, I, O) \leftarrow trie\_property(T, alias(dual)), dual(T, N, P, I, O).$

2. $dual(T, N, P, I, O) \leftarrow trie\_interned(d(N, P, Dual, \_), T),$
   $call\_dual(P, I, O, Dual).$
3. $dual(T, N, P, I, O) \leftarrow current\_pos(T, N, P, Pos),$
   $dualize(Pos, Dual, NextPos),$
   $store\_dual(T, N, P, Dual, NextPos),$
   $call\_dual(P, I, O, Dual).$

Assuming that a trie T with alias $dual$ has been created, predicate $dual/4$ (line 1) is defined by an auxiliary predicate $dual/5$ with an access to the trie T, the access being provided by the trie manipulation predicate $trie\_property/2$. Lines 2 and 3 give the definition of $dual/5$:

- In the first definition (line 2), an attempt is made to reuse generic dual rules, which are stored already as facts $d/4$ in trie T. This is accomplished by unifying terms in T with $d(N, P, Dual, \_)$, one at a time through backtracking, via the trie manipulation predicate $trie\_interned/2$. Predicate $call\_dual/4$ then does the job as before.

- The second definition (line 3) constructs generic dual rules lazily. It finds, via $current\_pos/4$, the current position Pos of the literal from the N-th rule of P, which can be obtained from the last argument of fact $d(N, P, Dual, Pos)$ stored in trie T. Using this Pos information, a new generic dual rule $Dual$ is constructed by means of

dualize/3. The predicate *dualize*/3 additionally updates the position of the literal, *NextPos*, for the next dualization. The dual rule *Dual*, together with the tracking information, is then memoized as a fact $d(N, P, Dual, NextPos)$ in trie T, via *store_dual*/5. Finally, the just constructed dual *Dual* is instantiated and invoked using *call_dual*/4.

Whereas the first approach constructs generic dual rules by need eagerly, the second one does it lazily. But this requires memoizing dual rules to be carried out explicitly, and additional tracking information is needed to correctly pick up on dual rule generation at the point where it was last left. This approach affords us a simulation of batched table scheduling for *dual*/5, within the default local table scheduling.

5.1.3 Concluding Remarks

TABDUAL is an ongoing work, which primarily intends to sensitize a general audience of users, and of implementers of various LP systems, to the potential benefits of tabling in abduction. TABDUAL has been evaluated with various evaluation objectives (Saptawijaya and Pereira, 2015):

- We evaluate the benefit of tabling abductive solutions, where we employ an example from declarative debugging to debug missing solutions of logic programs, via a process now characterized as abduction (Saptawijaya and Pereira, 2013b), instead of as belief revision (Pereira et al., 1993a; Pereira et al., 1993b).
- We use the other case of declarative debugging, that of debugging incorrect solutions, to evaluate the relative worth of the dual transformation by need.
- We touch upon tabling abductive solution candidates that violate constraints, the so-called *nogoods* of subproblems in the context of abduction, and show that tabling abductive solutions can be appropriate for this purpose.
- We also evaluate TABDUAL in dealing with programs having loops, where we compare its results with those obtained from an implementation of ABDUAL (Alferes et al., 2004b).

Other implementation aspects of TABDUAL, such as dealing with programs having loops and dynamically accessing ongoing abductive solutions for preferring amongst them, as well as the evaluation of TABDUAL, are discussed at length in Saptawijaya and Pereira (2015). They are beyond the scope of this thesis.

TABDUAL still has much room for improvement. Future work will consist in continued exploration of our applications of abduction, which will provide feedback for system improvement. Tabled abduction may benefit from answer subsumption (Swift and Warren, 2010) in tabling abductive solutions to deal with redundant explanations, in the sense that it suffices to table only smaller abductive solutions (with respect to the subset relation).

Another potential XSB Prolog's feature to look into is the applicability of interning ground terms (Warren, 2013) for tabling abductive solutions, which are ground, and study how extra efficiency may be gained from it.

The implementation technique of BY-NEED(LAZY) consists in operational details that are facilitated by XSB Prolog's trie manipulation predicates, to simulate the batched-like table scheduling within XSB Prolog's current default local table scheduling. In order to have a more transparent implementation of those operations, it is desirable that XSB Prolog permit a mixture in using batched and local table scheduling strategies, or alternatively, stopping the evaluation at some first answers to a subgoal within the currently default local table scheduling.

Though TABDUAL is implemented in XSB Prolog, a number of its techniques are adaptable and importable into other LP systems that afford required tabling mechanisms. They add and aid to the considerations involved in the research of the still ongoing developments of tabling mechanisms in diverse LP systems, and serve to inspire these systems in terms of solutions, options and experimentation results of incorporating tabled abduction.

5.2 Incremental Tabling of Fluents for LP Updating

Incremental tabling (Saha, 2006; Swift, 2014), available in XSB Prolog, is an advanced recent tabling feature that ensures the consistency of answers in a table with all dynamic clauses on which the table depends. It does so by incrementally maintaining the table, rather than by recomputing answers in the table from scratch to keep it updated. The applications of incremental tabling in LP have been demonstrated in pointer analyses of C programs in the context of incremental program analyses (Saha and Ramakrishnan, 2005), data flow analyses (Saha and Ramakrishnan, 2006), static analyses (Eichberg et al., 2007), incremental validation of XML documents and push down model checking (Saha, 2006). This range of applications suggests that incremental tabling lends itself to dynamic environments and evolving systems, including notably logic program updating.

We conceptualize a technique with incremental tabling that permits a reconciliation of high-level top-down deliberative reasoning about a goal, with autonomous low-level bottom-up world reactivity to ongoing updates. The technique, dubbed EVOLP/R, is theoretically based on Dynamic Logic Programs (Alferes et al., 2000) and its subsequent development, Evolving Logic Programs (EVOLP) (Alferes et al., 2002a).

5.2.1 The EVOLP/R Language

The syntax of EVOLP/R adopts that of *generalized logic programs*, following the definitions in Alferes et al. (2000), which is also the basis of the EVOLP language (Alferes et al., 2002a). We leave out EVOLP's reserved predicate *assert*/1 in the EVOLP/R language, which is

enough for our purpose in this thesis. Moreover, updates are restricted to fluents only, which is explained below.

Generalized logic programs allow negative information to be represented in logic programs and in their updates, due to the possibility to have a default negation not only in the body of a rule but also in its head. As in Alferes et al. (2000), for convenience, generalized logic programs are *syntactically* represented as propositional Horn theories. In particular, default negation *not A* is represented as a standard propositional atom *not_A*.

Let $\mathcal{K}$ be an arbitrary set of propositional atoms whose names do not begin with a "*not_*". The propositional language $\mathcal{L}_\mathcal{K}$ *generated* by $\mathcal{K}$ is the language whose set of propositional atoms consists of:

$$\{A : A \in \mathcal{K}\} \cup \{not\_A : A \in \mathcal{K}\}.$$

Definition 22 (Generalized Logic Program) *A* generalized logic program *over the language $\mathcal{L}_\mathcal{K}$ is a countable set of rules of the form:*

$$L \leftarrow L_1, \ldots, L_n$$

where L and L_is are atoms from $\mathcal{L}_\mathcal{K}$.

The evolution of a program is formed through a sequence of program updates. As in EVOLP, the sequence of programs are treated as in Dynamic Logic Programs (DLPs). We denote a set of *states* as $\mathcal{S} = \{1, 2, \ldots, s, \ldots\}$ of natural numbers. Let $\mathcal{P} = \{P_i : i \in \mathcal{S}\}$. A *dynamic logic program* $\bigoplus_s \mathcal{P}$ is a sequence of programs $P_1 \oplus \cdots \oplus P_s$. If the set S has the largest element max, we simply write $\bigoplus \mathcal{P}$ instead of $\bigoplus_{max} \mathcal{P}$.

In EVOLP/R, the evolution of a program P of the language $\mathcal{L}_\mathcal{K}$ from the initial state 1 to a state s is specified as a dynamic logic program $\bigoplus_s \mathcal{P}$, where:

- $P = P_1$ (the program P is given at the initial state 1).
- For $2 \leq i \leq s$, P_i is a set of atoms from $\mathcal{L}_\mathcal{K}$, referred as *fluents* (i.e., state-dependent atoms).

The *negation complement of a fluent A* is denoted by $compl_{\text{FL}}(A)$, where the complement of atom A and its negation *not_A* is defined as $compl_{\text{FL}}(A) = not\_A$ and $compl_{\text{FL}}(not\_A) = A$, respectively.

Next, we define the semantics of a DLP in EVOLP/R. As we consider a propositional language $\mathcal{L}_\mathcal{K}$, we first adapt Definition 2 of two-valued interpretations. Let $F \subseteq \mathcal{K}$ and *not F* represent $\{not\_A : A \in F\}$. A two-valued interpretation M over $\mathcal{L}_\mathcal{K}$ is a set of atoms $T \cup not\ F$ from $\mathcal{L}_\mathcal{K}$, such that $T \cup F = \mathcal{K}$ and $T \cap F = \emptyset$.

Definition 23 *Let $\bigoplus\{P_i : i \in \mathcal{S}\}$ be a dynamic logic program over language $\mathcal{L}_\mathcal{K}$, $s \in S$, and M be a two-valued interpretation over $\mathcal{L}_\mathcal{K}$. Then:*

$$Default_s(M) = \{not\_A \mid \nexists A \leftarrow Body \in P_i,\ 1 \leq i \leq s,\ M \models Body\}$$
$$Reject_s(M) = \{A \leftarrow Body \in P_i \mid \exists\, compl_{\text{FL}}(A) \leftarrow Body' \in P_j, i < j \leq s \wedge M \models Body'\}$$

where both Body and Body′ are conjunctions of literals.

Observe that since updates in EVOLP/R are restricted to fluents only, $M \models Body'$ in the definition of $Reject_s(M)$ is vacuously true, as we have an empty $Body'$ for an updating fluent $compl_{\mathrm{FL}}(A)$.

Definition 24 *A two-valued interpretation M over $\mathcal{L}_\mathcal{K}$ is a stable model of a dynamic logic program $\bigoplus \{P_i : i \in \mathcal{S}\}$ at state $s \in \mathcal{S}$ iff:*

$$M = least\left(\left[\bigcup_{i \leq s} P_i - Reject_s(M)\right] \cup Default_s(M)\right)$$

The emphasis of EVOLP/R is on the implementation technique to demonstrate an innovative use of tabling, particularly the incremental tabling afforded by XSB Prolog, for dynamic program updating. Whereas XSB Prolog computes the Well-Founded Semantics, the semantics of dynamic logic programs in Definition 24 is based on the stable model semantics due to its simplicity.[2] Moreover, EVOLP/R currently considers only stratified programs (programs with no loops over negation), and the semantics for such programs therefore consists of only one stable model, which is also the well-founded model. This constraint is deliberately so made, at this point, as we want to concentrate on the incremental tabling aspects and usage for logic program updating, and later in its combination with abduction. The usage of incremental tabling for non-stratified programs within EVOLP/R, viz., for updating conditional answers, is a future line of work.

5.2.2 Incremental Tabling

Whenever a tabled predicate depends on dynamic predicates and the latter are updated (with Prolog's assert or retract predicates), these updates are not immediately reflected in the table, i.e., the table becomes out of date. This problem is known as the view maintenance problem in databases and the truth maintenance problem in artificial intelligence.

In "classical" tabling, a typical solution to this problem is to rely on the user to explicitly abolish the table whenever a dynamic predicate, on which the table depends, is updated. As several updates may take place on a dynamic predicate, such explicit table abolishment is rather inconvenient and defeats the benefit of tabling itself, because those solutions in the abolished table have to be recomputed from scratch.

In order to overcome this problem, XSB Prolog allows maintaining particular tables incrementally, known as *incremental tabling*. That is, the answers in these tables are ensured to be consistent with all dynamic facts and rules upon which they depend. In XSB Prolog, this requires both tabled predicates and the dynamic predicates they depend on to be declared as incremental. By default, the current XSB Prolog (version 3.6) updates an incremental table *transparently* (Swift et al., 2015): after a sequence of updates Δ, an incremental table T that depends on Δ and all tables upon which T depends are automatically updated (if

[2]See Alferes et al. (1999) for the Well-Founded Semantics of generalized logic programs and Banti et al. (2004) for the Well-Founded Semantics of dynamic logic programs.

needed) whenever a future subgoal calls T. This transparent incremental tabling promotes a *lazy* updating strategy, where table T is marked as *invalid*, and when a subgoal calls this invalid table T (after the invalidation phase is completed), then T and any tables upon which T depends are recomputed to reflect the updates. On the other hand, if no calls are ever made to an invalid incremental table, it will never incur the cost of an update. This is in contrast with the eager updating strategy of the earlier XSB Prolog's version, in which invalidated tables are updated immediately.

Example 16 below demonstrates how incremental tabling is used.

Example 16 *We first consider the following program that does not use incremental tabling:*

$$r(X,Y) \leftarrow s(X,Y), Y < 4.$$
$$s(a,2).$$
$$s(b,4).$$

where $r/2$ *is a tabled predicate, declared in XSB Prolog as*

```
:- table r/2.
```

and $s/2$ *is a dynamic predicate, declared as*

```
:- dynamic s/2.
```

Query `?- r(X,Y)` *succeeds with* $X = a$ *and* $Y = 2$. *Suppose a new fact* $s(c,1)$ *is asserted, via a usual Prolog assertion* `assert(s(c,1))`. *The answer of the same query* `?- r(X,Y)` *has not changed. The new solution* $X = c$ *and* $Y = 1$ *is not part of the query's solution, because the table is already created and the second invocation of the query just retrieves the existing single answer directly from the table.*

With incremental tabling, XSB Prolog automatically keeps the table for $r/2$ *correct with respect to the given update, returning also the new answer* $X = c,\ Y = 1$. *This is achieved by declaring* $r/2$ *as an incremental tabled predicate:*

```
:- table r/2 as incremental.
```

and $s/2$ *as an incremental dynamic predicate:*

```
:- dynamic s/2 as incremental.
```

Moreover, a specific incremental assertion `incr_assert(s(c,1))` *should be used for incremental tabling instead of a usual* `assert(s(c,1))`.

The reader is referred to Swift et al. (2015) for the further examples, predicates, and features of incremental tabling.

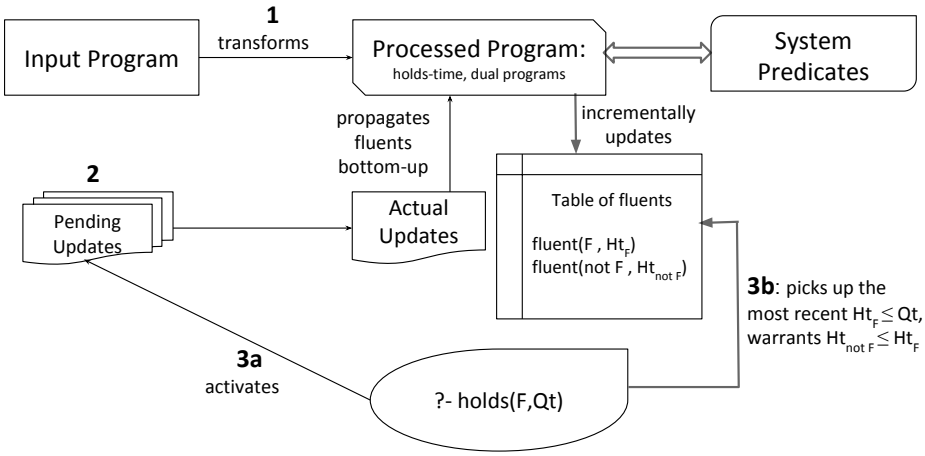


Figure 5.2: The concept of the EVOLP/R approach

5.2.3 The EVOLP/R Approach

Since incremental tabling allows tables to be correctly updated according to changes made by incremental assertions (or retractions), possibly in a chain of dependencies between tabled predicates, this feature can be exploited for automatically propagating fluent updates while tabling the states that indicate when a fluent holds true. By tabling this state information the recursive nature of the inertia principle can be avoided, as tabling readily provides the history of the world, and incremental tabling maintains this history to reflect changes made to the world.

The concept of the EVOLP/R approach is depicted in Figure 5.2 and is summarized below:

Step 1 A program transformation is first applied to transform the input program into the processed one. The transformation is responsible for, amongst others, adding extra information for rules (e.g., unique rule names, information about states, incremental table and dynamic declaration), providing an interface to EVOLP/R system predicates, and adding dual rules (similar to those used in TABDUAL, see Definition 18) to help propagate the dual negation complement of a fluent incrementally, to be explained below.

Step 2 The issued fluent updates are initially kept pending in the database.

Step 3 Query $?- \ holds(F, Qt)$ is a top-goal query to test whether fluent F holds true at state Qt (referred to as *query-time*).

a) On the initiative of this query, i.e., by need only, incremental assertions make these pending updates become active, if not already so, but only those with states up to an actual Qt. Such assertions automatically trigger Prolog system-implemented incremental bottom-up propagation of fluent updates, recomputation of answers for affected tables.

b) After the table is correctly maintained, this query is answered by looking up a collection of states fluent F is true in the table, picking up the most recent one, and ensuring that its negation complement *not_F* with a later state does not exist in the table.

5.2.3.1 The EVOLP/R Transformation

We start with a simple example that illustrates what is required from the EVOLP/R transformation.

Example 17 *Let* $\mathcal{P} = \{P_1, P_2, P_3\}$, *where:*

$$\begin{array}{ll} P_1: & b \leftarrow a. \\ & c \leftarrow b. \\ P_2: & a. \\ P_3: & not\_a. \end{array}$$

Program P_1 evolves from the initial state 1 through a series of fluent updates: it is updated at state 2 with fluent a, and at state 3 with fluent *not_a*. We introduce an incremental tabled predicate $fluent(F, T)$ to incrementally record that fluent F is true at a state T. The update of fluent a in P_2 (at state $i = 2$) is accomplished via the XSB Prolog's $incr\_assert/1$ system predicate, viz., by the incremental assertion $incr\_assert(a(2))$ to say that fluent a is incrementally asserted at state $i = 2$. Such an incremental assertion results in having entry $fluent(a, 2)$ in the table. Due to the dependencies of the three fluents, as defined by the two rules in P_1, the incremental assertion of a propagates to fluents b and c, leading to tabling $fluent(b, 2)$ and $fluent(c, 2)$. We thus have $fluent(a, 2)$, $fluent(b, 2)$, and $fluent(c, 2)$ that the three fluents are true at state $i = 2$, which conforms with the stable model $M_2 = \{a, b, c\}$ of $\bigoplus_2 \mathcal{P}$.

A subsequent update of fluent *not_a* at state $i = 3$ via $incr\_assert(not\_a(3))$ results in tabling $fluent(not\_a, 3)$. That means, we still have all previous tabled entries, viz., $fluent(a, 2)$, $fluent(b, 2)$, and $fluent(c, 2)$, plus now $fluent(not\_a, 3)$. While a is no longer true at this state ($i = 3$), which now can easily be checked by comparing states of a in the table (fluent a at $i = 2$ is supervened by its complement *not_a* at a later state $i = 3$), the same reasoning cannot be applied to fluents b and c. Different from before, the incremental assertion of *not_a* unfortunately does not propagate to tabling fluents *not_b* an *not_c*, i.e., neither $fluent(not\_b, 3)$ nor $fluent(not\_c, 3)$ found in the table. Indeed, there are no corresponding rules in P for *not_b* and *not_c*; thus failing to conclude that both fluents are also false at $i = 3$ by propagating *not_a* (cf. $M_3 = \{not\_a, not\_b, not\_c\}$ of $\bigoplus_3 \mathcal{P}$). In order

to provide rules for *not_b* and *not_c*, we adopt the dual program transformation, which is similar to that we use in TABDUAL, see Definition 18, but shown below by unfolding the first and second layers dual rules:

$$not\_b \leftarrow not\_a.$$
$$not\_c \leftarrow not\_b.$$

The introduced dual rules now allow the propagation from *not_a* to *not_b* and then to *not_c*, resulting in having $fluent(not\_b, 3)$ and $fluent(not\_c, 3)$ in the table. By having the latter two entries in the table, using the same previous reasoning, it can be concluded that fluents b and c are also false at $i = 3$, confirming M_3 of $\bigoplus_3 \mathcal{P}$.

This example hints at important information to be added in the transformation:

1. *Timestamp* that corresponds to state and serves as the only extra argument of fluents. It denotes the state when a fluent becomes true, referred to as *holds-time*.
2. *Dual rules* that are obtained using the transformation based on Definition 18.

The transformation technique is illustrated by Example 18 below, and is explained subsequently.

Example 18 *Recall program P_1:*

$$b \leftarrow a.$$
$$c \leftarrow b.$$

which transforms into:

1. $\#r(b, [a], 1).$
2. $\#r(c, [b], 1).$
3. $b(H) \leftarrow fluent(\#r(b, [a]), H_r),\ fluent(a, H_a),\ latest([(\#r(b, [a]), H_r), (a, H_a)], H).$
4. $not\_b(H) \leftarrow fluent(not\_\#r(b, [a]), H).$
5. $not\_b(H) \leftarrow fluent(not\_a, H).$
6. $c(H) \leftarrow fluent(\#r(c, [b]), H_r),\ fluent(b, H_b),\ latest([(\#r(c, [b]), H_r), (b, H_b)], H).$
7. $not\_c(H) \leftarrow fluent(not\_\#r(c, [b]), H).$
8. $not\_c(H) \leftarrow fluent(not\_b, H).$
9. $not\_a(1).$

In EVOLP/R, the initial timestamp is set at 1, when a program is inserted. Fluent predicates can be defined as facts (extensional) or by rules (intensional). In Example 18, both fluents b and c are defined intensionally. For such rule regulated intensional fluent instances, unique rule name fluents are introduced. A *rule name* fluent is a special fluent $\#r(H, [B])$,

which uniquely identifies the rule $H \leftarrow B$, and is introduced in its body, for controlling the activation of the rule (see Poole (1988)). In Example 18, rule name fluents $\#r(b,[a])$ and $\#r(c,[b])$ are introduced for rules $b \leftarrow a$ and $c \leftarrow b$, respectively. They are extensional fluent instances, and like any other extensional fluent instances, such a rule name fluent is translated by adding an extra argument (the third one) that corresponds to its holds-time (lines 1 and 2). In this case, each rule name fluent is true at the initial time 1, viz., the time when its corresponding rule is inserted.

Line 3 shows the translation of rule $b \leftarrow a$ of the input program. The single extra argument in its head is its holds-time, H. The body, which originally comprises just a call to goal a, is translated into two calls wrapped with the EVOLP/R's reserved *incremental tabled* predicate $fluent/2$ (defined in Section 5.2.3.2), viz., $fluent(\#r(b,[a]),H_r)$ and $fluent(a,H_a)$, which provides holds-time H_r and H_a of fluents $\#r(b,[a])$ and a, respectively. Note the inclusion of the unique rule name fluent (i.e., the call $fluent(\#r(b,[a]),H_r)$) in the body, whose purpose is to switch the corresponding rule on or off; this being achieved by asserting the rule name fluent or its negation complement, respectively. The holds time H of fluent b in the head is thus determined by which inertial fluent in its body holds the latest, via the EVOLP/R's $latest/2$ reserved predicate (defined in Section 5.2.3.2), which also assures that no fluents in the body were subsequently supervened by their complements at some time before H.

Lines 4 and 5 show the dual rules for b in their flattened form (unfolding the first and second layers dual rules). Line 4 expresses how the negation complement $not\_\#r(b,[a])$ of rule name fluent $\#r(b,[a])$ propagates to fluent not_b, whereas line 5 expresses the other alternative: how the negation complement not_a of fluent a propagates to fluent not_b.

Similar technique is applied to rule $c \leftarrow b$, resulting in rules shown in lines 6-8. Finally, line 9 is the dual rule for atom a that has no rule in P_1.

Since every fluent occurring in the program is subject to updates, all fluents and their negation complements should be declared as dynamic and incremental; the latter attribute is due to incremental tabling of fluents by the incremental tabled predicate $fluent/2$. For Example 18, we have, e.g.,

```
:- dynamic a/1, not_a/1 as incremental.
```

and similarly for fluents b, c, $\#r(b,[a])$, $\#r(c,[b])$, as well as their negation complements.

5.2.3.2 Reserved Predicates

Predicate $fluent(F,T)$ used in the transformation is a tabled one. Its dependency on fluent F, which is dynamic incremental, indicates that $fluent/2$ is tabled *incrementally*, and therefore declared

```
:- table fluent/2 as incremental.
```

This predicate is and defined as follows:

$$fluent(F, T) \leftarrow extend(F, [T], F'),\ call(F').$$

where $extend(F, Args, F')$ extends the arguments of fluent F with those in list $Args$ to obtain F'.

Updates propagation in EVOLP/R is query-driven, within some query-time of interest. This means we can use the given query-time to control updates propagation by keeping the sequence of updates pending in the database, and then making active, through incremental assertions, only those with the states up to the actual query-time (if they have not yet been so made already by queries of a later time stamp).

For expressing pending updates, we introduce a dynamic predicate $pending(F, T)$ to indicate that update of fluent F at state T is still pending, and use Prolog $assert/1$ predicate, i.e., $assert(pending(F, T))$, to assert such a pending fluent update into the Prolog database. Activating pending updates (up to the actual query-time Qt), as shown by the code below, can thus be done by calling all $pending(F, T)$ facts with $T \leq Qt$ from the database and actually asserting them incrementally using the XSB Prolog's system predicate $incr\_assert/1$:

$$\begin{array}{lcl} activate\_pending(Qt) & \leftarrow & pending(F, T),\ T \leq Qt,\ extend(F, [T], F'), \\ & & incr\_assert(F'),\ retract(pending(F, T)), fail. \\ activate\_pending(\_). & & \end{array}$$

We have seen predicate $latest([(F_1, H_1), \ldots, (F_n, H_n)], H)$ in the transformation, which appears in the body of a rule transform, say of fluent F. This reserved predicate is responsible for obtaining the latest holds-time H of F amongst fluents $F_1, \ldots, F_n$ in the body, while also assuring that none of them were subsequently supervened by their complements at some time up to H. It is defined as:

$$latest(Fs, H) \leftarrow greatest(Fs, H),\ not\_supervened(Fs, H).$$

where $greatest(Fs, H)$ extracts from list Fs, of (F_i, H_i) pairs with $1 \leq i \leq n$, the greatest holds time H among the Hi's. The predicate $not\_supervened(Fs, H)$ subsequently guarantees that there is no fluent complement F_i' (with holds time H_i') of F_i in Fs, such that $H_i < H_i' \leq H$.

Finally, the top-goal query $holds(F, Qt)$ in EVOLP/R (see Figure 5.2) is defined below. This reserved predicate is introduced to test whether fluent F is true at query-time Qt by first activating pending updates up to Qt:

$$\begin{array}{lcl} holds(F, Qt) & \leftarrow & activate\_pending(Qt),\ compl(F, F'), \\ & & most\_recent(F, H_F, Qt),\ most\_recent(F', H_{F'}, Qt), \\ & & H_F \neq 0,\ H_F \geq H_{F'}. \end{array}$$

where the predicate $compl(F, F')$ obtains the fluent negation complement $F' = compl_{\mathrm{FL}}(F)$, and the predicate $most\_recent(F, H_F, Qt)$ calls $fluent(F, H)$ and picks up the entry of

fluent F from the table with the highest timestamp $H_F \leq Qt$ (or returns $H_F = 0$, if the call $fluent(F, H)$ fails). For fluent F to hold true in the query-time Qt, this definition warrants that F ($H_F \neq 0$, for F not to trivially fail) is not supervened by its complement F', i.e., $H_F \geq H_{F'}$, where H' is obtained by invoking $most\_recent(F', H_{F'}, Qt)$.

5.2.4 Concluding Remarks

The EVOLP/R approach is somewhat similar and complementary approach to Logic-based Production System (LPS) with abduction (Kowalski and Sadri, 2011). The latter aims at defining a new logic-based framework for knowledge representation and reasoning, relying on the fundamental role of state transition systems in computing, and involving fluent updates by *destructive* assignment. In EVOLP/R, fluent updates are not managed by destructive database assignments, but rather tabled, thereby allowing to inspect their truths at a particular time, e.g., querying the past, which is important in counterfactual reasoning, as we discuss in Chapter 6.

Our first approach of EVOLP/R (Saptawijaya and Pereira, 2013a) preliminarily exploits the combination of incremental tabling and *answer subsumption* (Swift and Warren, 2010). The latter allows tables to retain only answers that subsume others with respect to some order relation. In that first approach, answer subsumption of fluent literals aims at addressing the frame problem, i.e., by automatically keeping track of only their *latest* assertion with respect to a given query-time.

The combined use of incremental tabling and answer subsumption is realized in the incrementally tabled predicate $fluent(F, Ht_F, Qt)$ for fluent literal F, where Ht_F and Qt are the holds-time of F and the query-time, respectively. Note the extra argument Qt in this specification of $fluent/3$. Invoking $fluent(F, Ht_F, Qt)$ thus, either looks for an entry in its table, if one exists; otherwise, it invokes dynamic definitions of fluent F, and returns *the* latest holds-time Ht_F with respect to a given query-time Qt. In order to return only the latest holds-time Ht_F (with respect to Qt), $fluent/3$ is tabled using answer subsumption on its second parameter, which is declared in XSB Prolog as:

```
:- table fluent(_,po('>'/2),_) as incremental.
```

to mean that only those answers that are maximal according to the partial order $>/2$ (arithmetic greater-than comparison relation) are tabled. In terms of $fluent/3$, it returns the latest holds-time (the second parameter in which the answer subsumption is applied) within a given query-time.

While answer subsumption is shown useful in this approach to avoid recursing through the frame axiom by allowing direct access to the latest time when a fluent is true, it requires $fluent/3$ to have query time Qt as its argument. Consequently, it may hinder the reusing of tabled answers of $fluent/3$ by similar goals which differ only in their query-time. Ideally, the state of a fluent literal in time depends solely on the changes made to the world, and not on whether that world is being queried. As an illustration,

suppose $fluent(a, 2, 4)$ is already tabled, and fluent a is inertially true till it is supervened by its negation complement not_a, say at time $T = 7$. When a new goal $fluent(a, Ht, 5)$ is posed, it cannot reuse the tabled answer $fluent(a, 2, 4)$, as they differ in their query time: the latter with $Qt = 4$, whereas the former $Qt = 5$. In this case, $fluent(a, Ht, 5)$ unnecessarily recomputes the same solution $Ht = 2$ (as fluent a is only supervened at $T = 7 > 5 = Qt$) and subsequently tables $fluent(a, 2, 5)$ as a new answer. A similar situation occurs when $fluent(a, Ht, 6)$ is queried, where $fluent(a, 2, 6)$ is eventually added into the table. This is clearly superfluous, as existing tabled answers could actually be reused and such redundancies avoided, if the tabled answers are independent of query time.

The above issue is addressed by our approach detailed in Section 5.2, where the use of incremental tabling in EVOLP/R is fostered further, while leaving out the problematic use of answer subsumption. The main idea, not captured in the first approach, is the perspective that knowledge updates (either self or world wrought changes) occur whether or not they are queried: the former take place independently of the latter, i.e., when a fluent is true at Ht, its truth lingers on independently of Qt. Consequently, from the standpoint of the tabled $fluent$ predicate definition, Qt no longer becomes its argument: we now have incremental tabled predicate $fluent(F, Ht)$.

The current EVOLP/R approach can also be extended by considering the reserved predicate $assert/1$ (not to be confused with the Prolog predicate $assert/1$) into its language, as introduced in EVOLP (Alferes et al., 2002a). By having $assert(F)$ in the head of a rule, the program is updated by fluent F, whenever the assertion $assert(F)$ is true in a model; or retracts F in case $assert(not\_F)$ obtains in the model under consideration. However, the EVOLP/R transformation becomes more involved. For instance, consider the rule:

$$assert(a) \leftarrow b.$$

The transformation of this rule is illustrated below:

1. $assert(a, H) \leftarrow fluent(\#r(assert(a), [b]), H_r),\ fluent(b, H_b),$
   $latest([(\#r(assert(a), [b]), H_r), (b, H_b)], H).$
2. $a(H) \leftarrow fluent(assert(a), H_{as}),\ H \text{ is } H_{as} + 1.$
3. $not\_assert(a, H) \leftarrow fluent(not\_\#r(assert(a), [b]), H).$
4. $not\_assert(a, H) \leftarrow fluent(not\_b, H).$
5. $not\_b(1).$

This rule transforms into two rules. The rule in line 1 is obtained following the same transformation as before, by treating $assert(a)$ as a fluent. The rule in line 2 is derived as the effect of asserting a. That is, the truth of a is determined solely by the propagation of fluent $assert(a)$, indicated by the call $fluent(assert(a), H_{as})$. The holds time H of a is thus determined by $H_{as} + 1$, instead of H_{as}, because a is actually asserted one state ahead after the state at which $assert(a)$ holds.

The other rules in lines 3-5 are dual rules for fluent $assert(a)$ obtained similarly as before. Note that rule in line 2 does not produce any dual rule. From the semantics

viewpoint, once a is asserted, its truth remains intact by inertia till superseded, even if $assert(a)$ is retracted at a later time.

Introducing $assert/1$ into the language may require EVOLP/R to take care of non-termination of updates propagation. Consider program P below:

$$assert(not\_a) \leftarrow a.$$
$$assert(a) \leftarrow not\_a.$$

where a is true in a state s when a is asserted into the program, not_a is true in state $(s+1)$ as not_a is asserted subsequently, a is true again in state $(s+2)$, etc.; the evolution continues indefinitely. From the incremental tabling viewpoint, it indicates that a predefined upper time limit is required to delimit updates propagation, thereby avoiding infinite number of answers in the $fluent/2$ table. This requirement is realistic, as our view into the future may be bounded by some time horizon, analogous to bounded rationality. Such delimitation can be done via a predicate, say $upper\_time(Lim)$, indicating the predefined upper time limit Lim, and is called in the definition of $fluent/2$ to time-delimit their tabled answers, modified as shown below:

$$fluent(F,T) \leftarrow upper\_time(Lim),\ extend(F,[T],F'),\ call(F'),\ T \leq Lim.$$

In this example, when the fact $upper\_time(4)$ is given, the $fluent/2$ table will contain a finite number of answers concerning fluent a and its negation complement not_a, viz., $fluent(not\_a,1)$, $fluent(a,2)$, $fluent(not\_a,3)$, and $fluent(a,4)$.

The extension of the present EVOLP/R approach to deal with this assertion construct, and its issues that may arise from the use of incremental tabling, are to be explored in future.

Given our need to equip abduction with updating and that our TABDUAL technique affords tabling, the exploitation of incremental tabling for LP updating, as in EVOLP/R, anticipates the integration between the two reasoning features. To this end, we want to ensure that updates occurring in abduction should correctly maintain the table of abductive solutions incrementally, and incremental tabling provides this mechanism. As abduction in TABDUAL is accomplished by a top-down query-oriented procedure and the incremental tabling of fluents in EVOLP/R triggers bottom-up updates propagation, their integration thus permits top-down (deliberative) abduction to meet bottom-up (reactive) updates propagation, via tabling. We shall discuss this integration of LP abduction and updating with tabling in Section 7.3 of Chapter 7.

CHAPTER 6

COUNTERFACTUALS IN LOGIC PROGRAMMING

Counterfactuals capture the process of reasoning about a past event that did not occur, namely what would have happened had this event occurred; or, vice-versa, to reason about an event that did occur but what if it had not. An example, taken from Byrne (2007): *Lightning hits a forest and a devastating forest fire breaks out. The forest was dry after a long hot summer and many acres were destroyed*. One may think of a counterfactual about it, e.g., "if only there had not been lightning, then the forest fire would not have occurred".

In this chapter, we innovatively make use of LP abduction and updating in an implemented procedure for evaluating counterfactuals, taking the established approach of Pearl (2009) as reference. Our approach concentrates on pure non-probabilistic counterfactual reasoning in LP, resorting to abduction and updating, in order to determine the logical validity of counterfactuals under the Well-Founded Semantics. Nevertheless, the approach is adaptable to other semantics, e.g., Weak Completion Semantics (Hölldobler and Ramli, 2009) is employed in Pereira et al. (2015).[1] Though abstaining from probability, this approach may be suitable and applicable to instances when probabilities are not known or needed. Abstaining from probability permits focusing on the naturalized logic of human counterfactual moral reasoning, as discussed in Section 3.3. In particular, morality aims at definitive (not probable) conclusions of right and wrong. Moreover, people naturally do not compute formal probabilities, nor probabilities are always available, when making moral decisions via counterfactuals, though one can benefit from counterfactuals for inferring intentions through a probabilistic model to explain moral permissibility (Kleiman-Weiner et al., 2015). Note that, even though the LP technique introduced in this

[1] Both the Well-Founded Semantics (WFS) and the Weak Completion Semantics (WCS) are 3-valued semantics that differ in dealing with close world assumption (CWA) and rules with positive loops (e.g., $p \leftarrow p$). WFS enforces CWA, i.e., atom a that has no rule is interpreted as false, whereas in WCS undefined. Nevertheless, they can be transformed one to another: adding rules $a \leftarrow u$ and $u \leftarrow not\ u$ for a reserved atom u renders a unknown in WFS; alternatively, adding $a \leftarrow \bot$, where $\bot$ is false, enforces CWA in WCS. In this thesis, positive loops are not needed and do not appear throughout examples we consider.

chapter is relevant for modeling counterfactual moral reasoning, its use is general, not specific to morality.

Counterfactuals have been widely studied in philosophy (Collins et al., 2004; Halpern and Hitchcock, 2015; Lewis, 1973), psychology (Byrne, 2007; Epstude and Roese, 2008; Markman et al., 1993; McCloy and Byrne, 2000; Migliore et al., 2014; Roese, 1997), as well as from the computational viewpoint (Baral and Hunsaker, 2007; Ginsberg, 1986; Pearl, 2009; Pereira et al., 1991a; Vennekens et al., 2010). In Pearl (2009), counterfactuals are evaluated based on a probabilistic causal model and a calculus of intervention. Its main idea is to infer background circumstances that are conditional on current evidences, and subsequently to make a minimal required intervention in the current causal model, so as to comply with the antecedent condition of the counterfactual. The modified model serves as the basis for computing the counterfactual consequence's probability.

Instead of defining a new formalism for counterfactual reasoning, we adopt here Pearl's approach as an inspiration, but abstaining from probabilities – given the lack of pure non-probabilistic counterfactual reasoning in LP – by resorting to LP abduction and updating. LP lends itself to Pearl's causal model of counterfactuals. For one, the inferential arrow in a LP rule is adept at expressing causal direction. For another, LP is enriched with functionalities, such as abduction and defeasible reasoning with updates. They can be exploited to establish a LP non-probabilistic reconstruction of Pearl's counterfactuals evaluation procedure. That is, LP abduction is employed for discovering background conditions from observations made or evidences given, whereas defeasible logic rules allow achieving adjustments to the current model via hypothetical updates of intervention on the causal model.

We start, in Section 6.1, with a summary of Pearl's structure-based counterfactuals and how its main ingredients, viz., causation and intervention, can be captured in LP. We detail subsequently, in Section 6.2, our LP-based procedure to evaluate counterfactuals, and provide concluding remarks in Section 6.3.

6.1 Causation and Intervention in LP

Pearl (2009) proposes a structural theory of counterfactuals based on a probabilistic causal model (likened to a Causal Bayesian Network) and a calculus of intervention (viz., his do-calculus). A causal model M consists of two sets of variables U and V, and a set F of functions that decides how values are assigned to each variable $V_i \in V$. The variables in U are background knowledge that have no explanatory mechanism encoded in model M. The values of all variables in V are uniquely determined by every instantiation $U = u$ of the background knowledge.

Procedure 1 *Given evidence e, the probability of the counterfactual sentence "Y would be y had X been x" can be evaluated in a three-step process:*

1. ***Abduction**: Update the probability $P(u)$ by the evidence e to obtain $P(u \mid e)$. This step explains the past circumstance $U = u$ in the presence of evidence e.*

2. ***Action**: Modify M by the action $do(X = x)$. This step minimally adjusts model M by a hypothetical intervention via the external action $do(X = x)$ to comply with the antecedent condition of the counterfactual.*

3. ***Prediction**: Compute the probability $Y = y$ in the modified model. In this step the consequence of the counterfactual is predicted based on the evidential understanding of the past (Step 1), and the hypothetical modification performed in Step 2.*

In summary, the approach determines the probability of the counterfactual's consequence $Y = y$ by performing an intervention to impose the counterfactual's antecedent $X = x$ (other things being equal), given evidence e about $U = u$.

Two important constructions in Pearl's approach of counterfactuals are causal model and intervention. *Causation* denotes a specific relation of cause and effect. Causation can be captured by LP rules, where the inferential arrow in a logic rule represents causal direction. LP abduction is thus appropriate for inferring causation, providing explanation to a given observation. That said, LP abduction is not immediately sufficient for counterfactuals. Consider a simple logic program $P = \{b \leftarrow a\}$. Whereas abduction permits obtaining explanation a to observation b, the evaluation of counterfactual "if a had not been true, then b would not have been true" cannot immediately be evaluated from the conditional rule $b \leftarrow a$, for if its antecedent is false the counterfactual would be trivially true. That justifies the need for an *intervention*. That is, it requires explicitly imposing the desired truth value of a, and subsequently checking whether the predicted truth value of b consistently follows from this intervention. As described in Pearl's approach, such an intervention establishes a required adjustment, so as to ensure that the counterfactual's antecedent be met. It permits the value of the antecedent to differ from its actual one, whilst maintaining the consistency of the modified model. We resort to LP abduction and updating to express causal source and intervention, respectively.

6.1.1 Causal Model and LP Abduction

With respect to an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$, observation O corresponds to Pearl's definition for evidence e. That is, O has rules concluding it in program P, and hence does not belong to the set $\mathcal{AB}_{gL}$. Recall from Section 4.2 that $\mathcal{AB}_{gL}$ refers to the set of ground abducibles formed over the set of abducible predicates $\mathcal{AB}$.

In Pearl's approach, a model M consists of set U of background variables, whose values are conditional on case-considered observed evidences. These background variables are not causally explained in M, as they have no parent nodes in the causal diagram of M. In terms of LP abduction, they correspond to a set of abducibles $E \subseteq \mathcal{AB}_{gL}$ that provide abductive explanations to observation O. Indeed, these abducibles likewise have

no preceding causal explanatory mechanism, as they have no rules concluding them in the program.

In a nutshell, an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ that provides an abduced explanation $E \subseteq \mathcal{AB}_{gL}$ to the available observation O mirrors Pearl's model M with its specific U supporting an explanation to the current observed evidence e.

6.1.2 Intervention and LP Updating

Besides abduction, our approach benefits from LP updating, which allows a program to be updated by asserting or retracting rules, thus changing the state of the program. LP updating is appropriate for representing changes and dealing with incomplete information. The specific role of LP updating in our approach is twofold:

1. It updates the program with the preferred explanation to the current observation, thus fixing in the program the initial abduced background context of the counterfactual being evaluated.

2. It facilitates an apposite adjustment to the causal model by hypothetical updates of causal intervention on the program, by affecting defeasible rules in order to retain consistency.

Both roles are sufficiently accomplished by *fluent* (i.e., state-dependent literal) updates, rather than full-blown rule updates. In the first role, explanations are treated as fluents. In the second, reserved predicates are introduced as fluents for the purpose of intervention upon defeasible rules. For the latter role, fluent updates are particularly more appropriate than rule updates (e.g., intervention by retracting rules), because intervention is hypothetical only. Removing away rules from the program would be an overkill, as the rules might be needed to elaborate justifications and introspective debugging. Alternatively, rules can simply be switched off or on in time by means of rule name fluents mechanism, as demonstrated by EVOLP/R.

6.2 Evaluating Counterfactuals via LP Abduction and Updating

The procedure to evaluate counterfactuals in LP essentially takes the three-step process of Pearl's approach as its reference. That is, each step in the LP approach captures the same idea of its corresponding step in Pearl's.

In what follows, counterfactuals are distinguished from semifactuals (Byrne, 2007), as the LP procedure for the former is slightly different from the latter. The procedure for semifactuals will be discussed separately at the end of this section.

The key idea of evaluating counterfactuals with respect to an abductive framework, at some current state (discrete time) T, is as follows.

- In step 1, abduction is performed to explain the factual observation.[2] The observation corresponds to the evidence that both the antecedent and the consequence literals of the present counterfactual were factually false.[3] There can be multiple explanations available to an observation; choosing a suitable one among them is a pragmatic issue, which can be dealt with preferences or integrity constraints. The explanation fixes the abduced context in which the counterfactual is evaluated ("all other things being equal") by updating the program with the explanation.

- In step 2, defeasible rules are introduced for atoms forming the antecedent of the counterfactual. Given the past event E, that renders its corresponding antecedent literal false, held at factual state $T_E < T$, its causal intervention is realized by a hypothetical update H at state $T_H = T_E + \Delta_H$, such that $T_E < T_H < T_E + 1 \leq T$. That is, a hypothetical update strictly takes place between two factual states, thus $0 < \Delta_H < 1$. In the presence of defeasible rules, this update permits hypothetical modification of the program to consistently comply with the antecedent of the counterfactual.

- Finally, in step 3, the well-founded model of the hypothetical modified program is examined to verify whether the consequence of the counterfactual holds true at state T. One can easily reinstate to the current factual situation by canceling the hypothetical update, e.g., via a new update of H's complement at state $T_F = T_H + \Delta_F$, such that $T_H < T_F < T_E + 1$.

Based on these ideas and analogously to the three-step process of Pearl's, our approach is defined in Procedure 2, abstracting from the above state transition detail (cf. Section 7.3.2 for the implementation aspect of this state transition). The following definitions are needed by the procedure.

Definition 25 *A set of integrity constraint is* satisfied *in $WFM(P)$ iff none is false in $WFM(P)$. That is, the body of an integrity constraint is either false or undefined (Pereira et al., 1991b).*

We next rephrase Definition 15 about abductive solutions and relate them to explanations of observations. As our counterfactual procedure is based on the Well-Founded Semantics, the standard logical consequence relation $P \models F$ used in the definition below presupposes the Well-Founded Model of P in verifying the truth of formula F, i.e., whether F is true in $WFM(P)$.

Definition 26 *Given an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ and an observation O, a consistent abductive solution $E \subseteq \mathcal{AB}_{gL}$ is an* explanation *to observation O iff $P \cup E \models O$ and $\mathcal{IC}$ is*

[2] We assume that people are using counterfactuals to convey truly relevant information rather than to fabricate arbitrary subjunctive conditionals (e.g., "If I had been watching, then I would have seen the cheese on the moon melt during the eclipse"). Otherwise, implicit observations must simply be made explicit observations, to avoid natural language conundrums or ambiguities (Grice, 1991).

[3] This interpretation is in line with the corresponding English construct, cf. Hewings (2013), commonly known as *third conditionals*.

satisfied in $WFM(P \cup E)$, where all abducibles not appearing in E have been replaced by $\mathbf{u}$*, both in P and $\mathcal{IC}$.*[4]

Procedure 2 *Let $\langle P, \mathcal{AB}, \mathcal{IC} \rangle$ be an abductive framework, where program P encodes the modeled situation on which counterfactuals are evaluated. Consider a counterfactual:*

"If Pre had been true, then Conc would have been true"

where Pre and Conc are finite conjunctions of literals.

1. ***Abduction**: Let $compl(L)$ be the negation complement of a literal L. Compute an explanation $E \subseteq \mathcal{AB}_{gL}$ to the observation $O = O_{Pre} \cup O_{Conc} \cup O_{Oth}$, where:*

   - $O_{Pre} = \{compl(L_i) \mid L_i \text{ is in } Pre\}$;
   - $O_{Conc} = \{compl(L_i) \mid L_i \text{ is in } Conc\}$; *and*
   - O_{Oth} *is other (possibly empty) observations,*

   such that $O_{Oth} \cap (O_{Pre} \cup O_{Conc}) = \emptyset$.

   Update program P with E, obtaining program $P \cup E$.

2. ***Action**: For each literal L in conjunction Pre, introduce a pair of reserved meta-predicates $make(B)$ and $make\_not(B)$, where B is the atom in L.*

   These two meta-predicates are introduced for the purpose of establishing causal intervention. That is, they are used to express hypothetical alternative events to be imposed.

   This step comprises two stages:

   a) *Transformation:*

      - *Add rule $B \leftarrow make(B)$ to program $P \cup E$.*
      - *Add 'not $make\_not(B)$' to the body of each rule in P whose head is B. If there is no such rule, add rule '$B \leftarrow$ not $make\_not(B)$' to program $P \cup E$.*

      Let $(P \cup E)_\tau$ be the resulting transform.

   b) *Intervention:*
      Update program $(P \cup E)_\tau$ with literal $make(B)$ or $make\_not(B)$, for $L = B$ or $L =$ not B, respectively. Assuming that Pre is consistent, $make(B)$ and $make\_not(B)$ cannot be imposed at the same time.

      Let $(P \cup E)_{\tau,\iota}$ be the program obtained after these hypothetical updates of intervention.

3. ***Prediction**: Verify whether $(P \cup E)_{\tau,\iota} \models Conc$ and $\mathcal{IC}$ is satisfied in $WFM((P \cup E)_{\tau,\iota})$.*

This three-step procedure defines *valid* counterfactuals.

[4]This replacement of abducible $A \notin E$ with $\mathbf{u}$ in P and $\mathcal{IC}$ is an alternative but equivalent to adding $A \leftarrow \mathbf{u}$ into $P \cup E$, as foreseen by Definition 15.

Definition 27 *Let $\langle P, \mathcal{AB}, \mathcal{IC}\rangle$ be an abductive framework, where program P encodes the modeled situation on which counterfactuals are evaluated. The counterfactual*

"If Pre had been true, then Conc would have been true"

is valid *given observation $O = O_{Pre} \cup O_{Conc} \cup O_{Oth}$ iff O is explained by $E \subseteq \mathcal{AB}_{gL}$, $(P \cup E)_{\tau,\iota} \models Conc$, and $\mathcal{IC}$ is satisfied in $WFM((P \cup E)_{\tau,\iota})$.*

Since the Well-Founded Semantics supports top-down query-oriented procedures for finding solutions, checking validity of counterfactuals, i.e., whether their conclusion *Conc* follows (step 3), given the intervened program transform (step 2) with respect to the abduced background context (step 1), in fact amounts to checking in a derivation tree whether query *Conc* holds true while also satisfying $\mathcal{IC}$.

Example 19 *Recall the example from Byrne (2007):* Lightning hits a forest and a devastating forest fire breaks out. The forest was dry after a long hot summer and many acres were destroyed.

Let us slightly complicate it by having two alternative abductive causes for the forest fire, viz., storm (which implies lightning hitting the ground) or barbecue. Storm is accompanied by strong wind that causes the dry leaves falling onto the ground. Note that dry leaves are important for forest fire in both cases.

This example is expressed by an abductive framework $\langle P, \mathcal{AB}, \mathcal{IC}\rangle$, where P is the program below, $\mathcal{AB} = \{storm/0, barbecue/0\}$, and $\mathcal{IC} = \emptyset$:

$$\begin{aligned} fire &\leftarrow barbecue, dryLeaves. \\ fire &\leftarrow barbecue^*, lightning, dryLeaves, leavesOnGround. \qquad (6.1) \\ lightning &\leftarrow storm. \\ leavesOnGround &\leftarrow storm. \\ dryLeaves. & \end{aligned}$$

The use of $barbecue^*$ in the body of the rule 6.1 is intended so as to have mutual exclusive explanations. Consider counterfactual:

"If only there had not been lightning, then the forest fire would not have occurred"

where $Pre = not\ lightning$ and $Conc = not\ fire$.

1. ***Abduction**: Besides $O_{Pre} = \{lightning\}$ and $O_{Conc} = \{fire\}$, say that leavesOnGround is observed too: $O_{Oth} = \{leavesOnGround\}$. Given $O = O_{Pre} \cup O_{Conc} \cup O_{Oth}$, there are two possible explanations: $E_1 = \{storm, barbecue^*\}$ and $E_2 = \{storm, barbecue\}$.*

   Consider a scenario where the minimal explanation E_1 (in the sense of minimal positive literals) is preferred to update P, to obtain $P \cup E_1$. Note, program $P \cup E_1$ corresponds to a state with:

   $$WFM(P \cup E_1) = \{dryLeaves, storm, leavesOnGround, lightning, fire, not\ barbecue\}.$$

This updated program reflects the evaluation context of the counterfactual, where all literals of Pre and Conc were false in the initial factual situation.

2. ***Action**: The transformation results in program $(P \cup E_1)_\tau$:*

$$
\begin{aligned}
fire &\leftarrow barbecue, dryLeaves.\\
fire &\leftarrow barbecue^*, lightning, dryLeaves, leavesOnGround.\\
leavesOnGround &\leftarrow storm.\\
lightning &\leftarrow make(lightning).\\
lightning &\leftarrow storm, not\ make\_not(lightning).\\
dryLeaves.&
\end{aligned}
$$

Program $(P \cup E_1)_\tau$ is updated with $make\_not(lightning)$ as the required intervention, resulting in the program $(P \cup E_1)_{\tau,\iota}$ that corresponds to a new state with:

$$
\begin{aligned}
WFM((P \cup E_1)_{\tau,\iota}) = \{&dryLeaves, storm, leavesOnGround, make\_not(lightning),\\
&not\ make(lightning), not\ barbecue, not\ lightning,\ not\ fire\}.
\end{aligned}
$$

3. ***Prediction**: We verify that $(P \cup E_1)_{\tau,\iota} \models not\ fire$, and $I = \emptyset$ is trivially satisfied in $WFM((P \cup E_1)_{\tau,\iota})$.*

We thus conclude that, for this E_1 scenario, the given counterfactual is valid.

Example 20 *In the other explanatory scenario of Example 19, where E_2 (instead of E_1) is preferred to update P, the counterfactual is no longer valid, because:*

$$
\begin{aligned}
WFM((P \cup E_2)_{\tau,\iota}) = \{&dryLeaves, storm, leavesOnGround, barbecue, make\_not(lightning),\\
&not\ make(lightning), not\ lightning, fire\}
\end{aligned}
$$

and thus $(P \cup E_2)_{\tau,\iota} \not\models not\ fire$. Indeed, the forest fire would still have occurred but due to an alternative cause, viz., barbecue.

Skeptical and credulous counterfactual evaluations could ergo be defined, i.e., by evaluating the presented counterfactual for each abduced background context. Given that step 2 can be accomplished by a one-time transformation, such skeptical and credulous counterfactual evaluations require only executing step 3 for each background context fixed in step 1.

Semifactuals Reasoning

Another form related to counterfactuals is *semifactuals*, i.e., one that combines a counterfactual hypothetical antecedent and an unchanged factual consequence (Byrne, 2007), with a typical form of statement "Even if ...". Other comparable linguistic constructs also exist, e.g., "No matter if ...", "Though ..., ... still ...", etc. The LP procedure for counterfactuals

(Procedure 2) can easily be adapted to evaluating semifactuals. Like in counterfactuals, the antecedent of a semifactual is supposed false in the factual situation. But different from counterfactuals, the consequence of a semifactual should instead be factually ensured *true* (rather than false).

Consider the general semifactual form:

"Even if *Pre* had been true, *Conc* would still have been true".

Its LP evaluation follows Procedure 2 with the only modification on the definition of O_{Conc} in Step 1, i.e., for semifactuals, O_{Conc} is defined as $O_{Conc} = \{L_i \mid L_i \text{ is in } Conc\}$, to warrant its consequence factually true. The validity condition for semifactuals is the same as for counterfactuals, cf. Definition 27.

Example 21 *Recall Example 20, where $E_2 = \{storm, barbecue\}$ is preferred. Consider semifactual:*

"Even if there had not been lightning, the forest fire would still have occurred"

where $Pre = not\ lightning$ and $Conc = fire$.

This semifactual is valid, because given the same $WFM((P \cup E_2)_{\tau,\iota})$ as in Example 20, we now have $(P \cup E_2)_{\tau,\iota} \models Conc$, i.e., $(P \cup E_2)_{\tau,\iota} \models fire$.

6.3 Concluding Remarks

In Pearl's approach, intervention is realized by surface revision, by imposing the desired value to the intervened node and cutting it from its parent nodes. This is also the case in our approach, by means of hypothetical updates affecting defeasible rules that relate to the counterfactual's antecedent. Other subtle ways of intervention may involve deep revision, which can be realized in LP. It is beyond the scope of the thesis, but amply discussed in Pereira et al. (2015).

In Pereira et al. (2015), our procedure is reformulated using different semantics, viz., the weak completion semantics, and some counterfactual properties specific to our LP-based approach are discussed. Indeed, since the idea of each step in the LP approach mirrors the one corresponding in Pearl's, the LP approach therefore immediately compares to Pearl's, its epistemic adequacy and properties relying on those of Pearl's. The satisfaction of counterfactual properties, such as those logic properties of counterfactuals discussed in Lewis (1973), e.g., various fallacies that distinguish counterfactual conditional from material one, reflexive, modus tollens, disjunction in the antecedent, combination of sentences, etc., is not in the purview of the thesis, and left for future work.

LP abduction and revision are employed in Dietz et al. (2015) to evaluate indicative conditionals, but not counterfactual conditionals. LP abduction is employed through a rewrite system to find solutions for an abductive framework; the rewrite system intuitively captures the natural semantics of indicative conditionals. Rule revisions are additionally

used to satisfy conditions whose truth-value is unknown and which cannot be explained by abduction.

The study of counterfactual reasoning in Logic Programming is not new. In Pereira et al. (1991a), counterfactuals are evaluated using contradiction removal semantics of LP. The work is based on Lewis's counterfactuals (Lewis, 1973), where a model of a logic program represents a world in Lewis's concept. The semantics defines the most similar worlds by removing contradictions from the associated program, obtaining the so-called maximal non-contradictory submodels of the program. It does not concern itself with LP abduction and updating; both being relevant for our work, which is based on Pearl's concept rather than Lewis's, without the need of a world distance measure.

Probabilistic LP language P-log with the Stable Model Semantics is employed, in Baral and Hunsaker (2007), to encode Pearl's Probabilistic Causal Model, without involving abduction. It does not directly encode Pearl's three-step process, but focuses on P-log probabilistic approach to compute the probability of a counterfactual query. Our work does not deal with probability, but logic, though it epistemically mirrors Pearl's three-step process, via LP abduction and updating. Our approach is also not based on the stable model semantics, but instead on the Well-Founded Semantics with its relevancy property, which is more appropriate for LP abduction by need as argued earlier.

In Vennekens et al. (2010), Pearl's Probabilistic Causal Model is encoded using a different Probabilistic LP, viz., CP-logic, but without involving abduction either. Whereas P-log has its own *do*-operator to achieve intervention in its probabilistic reasoning, CP-logic achieves it by eliminating rules. Similar to P-log, our approach introduces meta-predicates *make* and *make_not* to accomplish intervention via defeasible rules and fluent updates, without eliminating rules, as CP-logic does.

Our procedure specifically focuses on evaluating counterfactuals in order to determine their validity. It is interesting to explore in future other aspects of counterfactual reasoning; some of them are identified below:

- We consider the so-called *assertive counterfactuals*, where a counterfactual is given as being a valid statement, rather than a statement whose truth validity has to be determined. The causality expressed by such a valid counterfactual may be useful for refining an existing knowledge base. For instance, suppose we have a rule stating that the lamp is on if the switch is on, written as $lamp\_on \leftarrow switch\_on$. Clearly, providing the fact *switch_on*, we have $lamp\_on$ true. Now consider that the following counterfactual is given as being a valid statement:

  "If the bulb had not functioned properly, then the lamp would not be on"

  There are two ways that this counterfactual may refine the rule about $lamp\_on$. First,

the causality expressed by this counterfactual can be used to transform the rule into:

$$lamp\_on \leftarrow switch\_on, bulb\_ok.$$
$$bulb\_ok \leftarrow not\ make\_not(bulb\_ok).$$

So, the lamp will be on if the switch is on – that is still granted – but subject to an update $make\_not(bulb\_ok)$, which captures the condition of the bulb. In the other alternative, an assertive counterfactual is rather directly translated into an update rule, and need not transform existing rules. If we consider an EVOLP-like updating language, the following rule update is enacted:

$$assert(not\ lamp\_on) \leftarrow not\ bulb\_ok$$

and the original rule $lamp\_on \leftarrow switch\_on$ can be kept intact. This rule update affects the query about $lamp\_on$ thereafter. Like before, the lamp will still be on if the switch is on, but now subject to a superseding $bulb\_ok$ update.

- We may extend the antecedent of a counterfactual with a rule, instead of just literals. For example, consider the following program (assuming an empty abduction, so as to focus on the issue):

$$warm\_blood(M) \leftarrow mammal(M).$$
$$mammal(M) \leftarrow dog(M).$$
$$mammal(M) \leftarrow bat(M).$$
$$dog(d). \qquad bat(b).$$

Querying $?-\ bat(B), warm\_blood(B)$ assures us that there is a warm blood bat, viz., $B = b$.

Now consider the counterfactual:

"If bats were not mammals they would not have warm blood".

Transforming the above program using our procedure obtains:

$$warm\_blood(M) \leftarrow mammal(M).$$
$$mammal(M) \leftarrow make(mammal(M)).$$
$$mammal(M) \leftarrow dog(M), not\ make\_not(mammal(M)).$$
$$mammal(M) \leftarrow bat(M), not\ make\_not(mammal(M)).$$
$$dog(d). \qquad bat(b).$$

The antecedent of the given counterfactual can be expressed as the rule:

$$make\_not(mammal(B)) \leftarrow bat(B).$$

We can check using our procedure that, given this rule intervention, the above counterfactual is valid: $not\ warm\_blood(b)$ is true in the intervened modified program.

- Finally, we can easily imagine the situation where the antecedent *Pre* of a counterfactual is not given, though the conclusion *Conc* is, and we want to abduce *Pre* in the form of interventions. That is, the task is to abduce *make* and *make_not*, rather than imposing them, while respecting the integrity constraints, such that the counterfactual is valid.

  Tabling abductive solutions, such as in TABDUAL, may be relevant in this problem. Suppose that we already abduced an intervention Pre_1 for a given $Conc_1$, and we now want to find Pre_2 such that the counterfactual "If Pre_1 and Pre_2 had been the case, then $Conc_1$ and $Conc_2$ would have been the case" is valid. In particular, when abduction is performed for a more complex conclusion $Conc_1$ and $Conc_2$, the solution Pre_1, which has already been abduced and tabled, can be reused in the abduction of such a more complex conclusion, leading to the idea that problems of this kind of counterfactual reasoning can be solved in parts or in a modular way.

In summary, this chapter presents a LP technique for evaluating counterfactuals by resorting to a combination of abduction and updating. It corresponds to the three-step procedure of Pearl's structural theory, omitting probability, and focuses on the logical validity of counterfactuals. In future, it is worth exploring possible extensions of this technique to the three counterfactual reasoning aspects discussed above.

CHAPTER 7

LOGIC PROGRAMMING SYSTEMS AFFORDING MORALITY EXPERIMENTS

In Chapter 4 we show the appropriateness of LP-based reasoning features for representing diverse issues of moral facets identified in Chapter 3. In this chapter, we discuss how these LP-based reasoning features are synthesized in three different systems: ACORDA (Section 7.1), PROBABILISTIC EPA (Section 7.2), and QUALM (Section 7.3). Whereas the development of QUALM is a contribution of this thesis, ACORDA (Lopes, 2006; Lopes and Pereira, 2006; Pereira and Lopes, 2009) and PROBABILISTIC EPA (Han, 2009; Han et al., 2008; Pereira and Han, 2009) are two existing systems that have been developed earlier, but not with any specific wide principled implementation of morality in mind, as we shall see.

Though these systems share its main feature, viz., abduction, each system concern itself with a particular combination of features. Moreover, their shared feature, abduction, implements different techniques, indicating the progress made in the development of these three systems. The three systems are employed to model, here, for the first time, different issues of considered moral facets, depending on the need of their respective combination of features. Their applications are elaborated in the subsequent Chapter 8.

7.1 ACORDA

ACORDA is a system that implements Prospective Logic Programming (Pereira and Lopes, 2009). Prospective Logic Programming enables an evolving program to look ahead prospectively into its possible future states and to prefer among them to satisfy goals. 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 LP abduction, updating, and preferences, in order to synthesize and maintain abductive hypotheses.

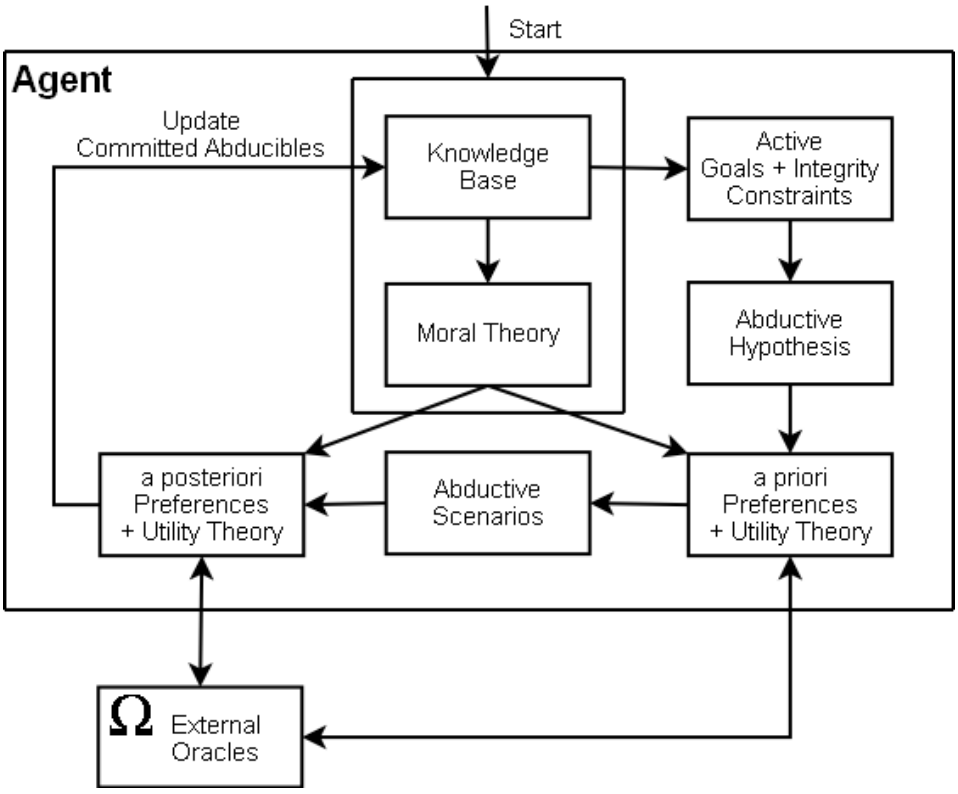


Figure 7.1: Prospective logic agent architecture (equipped with moral theory)

Figure 7.1 (Pereira and Lopes, 2009) 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 initial updatable state. The problem of prospection is then of finding abductive solutions to this initial and subsequent states which are both:

- *relevant*, i.e., under the agent's current goals; and
- *preferred*, i.e., with respect to preference rules in its knowledge base.

The first step is to select the goals that the agent will 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 domain-dependent preference rules, among available abducibles 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 some available moral theory to be defined. 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. The agent may use the additional information it acquires to commit to particular abductive solutions, which will be taken into account in a second round of prospection.

ACORDA implements an ad hoc abduction by means of even loops over negation, as described below, on top of an EVOLP meta-interpreter (Alferes et al., 2002b). Its implementation of preferences over abductive scenarios is supported by the results from Dell'Acqua and Pereira (2007).

ACORDA is implemented in XSB Prolog with its Well-Founded Semantics. Its implementation particularly benefits from the *XSB Answer Set Programming (XASP)* package (Castro et al., 2015). The XASP package extends the computation of the well-founded model by providing linkage to the stable model generator (answer set solver) *Smodels* (Niemelä and Simons, 1997) to compute two-valued models from the so-called *residual program* resulting from the top-down query-oriented procedure. The residual program corresponds to a *delay list*, which contains literals whose truth value is undefined in the well-founded model of a program. Such an integration with a stable model generator as provided by the XASP package allows maintaining the relevancy property in finding an answer to a query, by submitting only the relevant residual program to the stable model generator.

We discuss below the main constructs of ACORDA, which are adapted from Pereira and Lopes (2009).[1]

7.1.1 Active Goals

In each cycle of its evolution a prospective logic agent attends to a set of active goals. These active goals are triggered by using the observation $on\_observe(O, R, Q)$ construct. This construct generally refers to a relation amongst the observer O, the reporter R, the observation Q, and represents observations that may be reported by the environment to the agent, from one agent to another, or may also be from itself (self-triggered goals).

Being an active goal, triggering $on\_observe/3$ causes the agent to launch the query Q standing for the observations contained inside. In the prospection mechanism, when starting a cycle, the agent collects its active goals by finding all $on\_observe(agent, agent, Q)$ in the program, and subsequently computing abductive solutions, using the mechanism of abduction described below, for the conjunction of Qs obtained from all active goals, while also satisfying integrity constraints. Accordingly, if there is no such active goals in the program, the initial prospection mechanism amounts to satisfying integrity constraints only. In ACORDA, satisfying integrity constraints is realized by invoking the goal $not\ false$, where $false$ is the ACORDA's reserved atom for representing $\bot$ in the Definition 12 of integrity constraints.

[1] The initial version of ACORDA is based on Lopes (2006) and Lopes and Pereira (2006).

7.1.2 Abduction and a priori preferences

In prospective logic, only abducibles that are relevant for the problem in hand are generated. For so doing, the notion of expectation, as described in Section 4.3, is employed to express preconditions for enabling the assumption of an abducible, thus constraining a priori relevant abducibles with respect to the agent's actual situation.

An abducible A is only *considered* (and thereby abduced) if there is an expectation for it, and there is no expectation to the contrary. In ACORDA, this is represented as follows:

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

ACORDA implements an ad hoc abduction by means of even loops over negation for every positive abducible A:

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

As shown by Example 2 in Chapter 4, such representation causes the abducible A undefined in the Well-Founded Model of the program. It creates a relevant residual program in XSB Prolog with respect to the query derived from active goals (and integrity constraints). This relevant residual program can be sent to the stable models generator Smodels through the XASP package, as described above, which will returns abductive stable models that can be analyzed further through a posteriori preferences.

7.1.3 A Posteriori Preferences

Having computed abductive stable models, which correspond with possible scenarios, more favorable ones can be preferred *a posteriori*. A posteriori preferences are performed to reason about which consequences of abducibles, or other features of the models, are determinant for the final choice, reflecting the desired quality of the model.

One possibility of this a posteriori preferences reasoning is to consider a quantitative evaluation which can be based on different techniques of quantitative decision making. For instance, some measure of utility can be associated to each choice scenario, and the final choice is preferred by maximizing some utility function. Another possibility is to evaluate each choice scenario qualitatively, e.g., by enforcing this scenario to satisfy some properties.

When currently available knowledge of the situation is insufficient to commit to any single preferred abductive solution, additional information can be gathered, e.g., by performing experiments or consulting an oracle, in order to confirm or disconfirm some of the remaining candidates.

7.2 PROBABILISTIC EPA

ACORDA was further developed into the Evolution Prospection Agent (EPA) system (Han, 2009; Pereira and Han, 2009). Distinct from ACORDA, EPA considers a different

abduction mechanism with a posteriori preferences representation. EPA is subsequently extended with a new feature that allows probabilistic reasoning. The latter is based on the probabilistic logic programming language P-log (Baral et al., 2009) in a XSB implementation, named P-log(XSB) (Han et al., 2008). We refer to the this extension of EPA as PROBABILISTIC EPA.

The prospection mechanism in EPA is initiated in the same way as in ACORDA, viz., by collecting active goals when a cycle is started, and subsequently finding the abductive solutions of its conjunction, while satisfying integrity constraints. EPA simplifies the representation of an observed active goal G by $on\_observe(G)$, and uses the same reserved atom $false$ for the head of an integrity constraint.

We detail below the features of EPA that mainly distinguish itself from ACORDA.

7.2.1 Abduction and a priori preferences

In EPA, the reserved predicate $abds(L)$ is used for declaring abducibles (and their corresponding arity) in list L. Like ACORDA, EPA also employs the notion of expectation and considered abducibles for constraining relevant abducibles to the agent's actual situation. Nevertheless, its definition of *consider*/1 is slightly different:

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

where, distinct from ACORDA, the abduction of A is not enacted via an even loop over negation. Instead, the abduction mechanism in EPA is based on the dual program transformation of ABDUAL (Alferes et al., 2004a).

Distinct from ACORDA, whose implementation is based on the EVOLP meta-interpreter, the EPA system is implemented on top of the NEGABDUAL meta-interpreter (Alferes and Pereira, 2007). This meta-interpreter is based on ABDUAL (which is responsible for the abduction mechanism of EPA) but with an additional constructive negation feature. That is, in addition to the abduction mechanism provided by ABDUAL, NEGABDUAL also uses abduction for constructive negation, viz., by making the disunification predicate an abducible. This feature of constructive negation by abduction is beyond the scope of this thesis, but its discussion is referred to Ceruelo (2009). Given this implementation of EPA, for updating a program with new information, standard Prolog assertion and retraction predicates are used rather than the full EVOLP language (Alferes et al., 2002a).

7.2.2 A Posteriori Preferences

EPA introduces a specific syntax for a posteriori preferences:

$$A_i \ll A_j \leftarrow holds\_given(L_i, A_i),\ holds\_given(L_j, A_j). \tag{7.1}$$

where A_i, A_j are abductive solutions and L_i, L_j are literals representing consequences. Distinct from ACORDA, the a posteriori preferences in EPA does not make use of the the XASP package, but resorts instead to abduction itself. In particular, the a posteriori

preference rule 7.1 states that A_i is preferred to A_j if L_i and L_j are true as the consequence of abductive solutions A_i and A_j, respectively, without any further abduction being permitted. Optionally, the body of this preference rule may contain Prolog predicate for quantitatively comparing the consequences L_i and L_j.

Other specific evaluations may also figure in the body of the preference rule 7.1. For example, if the evaluation is based on expected utility maximization, then the preference rule can be expressed as:

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

The rule states that A_i is preferred to A_j if the expected utility value U_i of relevant A_i's consequences is greater than expected utility value U_j of A_j's. Other decision rules, such as maximizing the minimum gain (maximin) or minimizing the maximum possible loss (minimax), are also possible.

7.2.3 Probabilistic Reasoning

PROBABILISTIC EPA extends EPA with a probabilistic reasoning feature based on the probabilistic LP language P-log (Baral et al., 2009).

The original P-log implementation uses an answer set solver as a tool for computing stable models of its logical part. An alternative implementation of P-log in XSB Prolog, named P-log(XSB) (Han et al., 2008), uses the XASP package for interfacing with the answer set solver Smodels. As in ACORDA, this implementation of P-log in XSB Prolog with its underlying Well-Founded Semantics has the advantage of collecting only relevant abducibles for a given query, obtained by need via top-down search, while still benefiting from the computation of stable models through the XASP package. Moreover, the tabling mechanism in XSB Prolog may significantly improve the performance of P-log(XSB) compared to the original P-log implementation, as shown by the evaluation results in Han et al. (2008).

PROBABILISTIC EPA results from the integration of P-log(XSB) into EPA. We next summarize the components of P-log and their syntax in PROBABILISTIC EPA.

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 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.

Declaration 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

$c = \{x_1, \ldots, x_m\}$ 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:

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

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 rules, facts and integrity constraints, formed using literals of Σ.

Random Selection Rule This is a rule for attribute a, which has the form:

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

This means that the attribute instance $a(\bar{t})$ is random if the conditions in $Body$ are satisfied. The parameter $DynamicRange$ allows restricting the default range for random attributes. The parameter $RandomName$ 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 (called pa-atoms), which can be defined by the following pa-rule:

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

This rule expresses 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-atom is the head of some pa-rule, whose body is true. 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 outcome y of a random event $a(\bar{t})$. Statement $do(a(\bar{t}, y))$ indicates $a(\bar{t}) = y$ is enforced as the result of a deliberate action.

In a PROBABILISTIC EPA program, a P-log program is embedded by putting it between reserved keywords, beginPlog and endPlog. In PROBABILISTIC EPA, probabilistic information can be obtained using the P-log(XSB) reserved predicate $pr(Q, P)$ (Han et al., 2008), which computes the probability P of a given query Q. It can be embedded just like a

usual Prolog predicate, in any constructs of PROBABILISTIC EPA programs, including active goals, preferences, and integrity constraints. Moreover, since P-log(XSB) allows to code Prolog probabilistic meta-predicates, i.e., Prolog predicates that depend on $pr/2$ predicates, we can directly use this probabilistic meta-information in PROBABILISTIC EPA programs.

7.3 QUALM

In Chapter 5, we propose two original approaches that enjoy the benefit of tabling for different reasoning features, separately: one employs tabling in abduction (TABDUAL) and the other in updating (EVOLP/R). The integration of these two approaches is indispensable for our purpose of representing some moral facets, as argued in Chapter 4. This section discusses a unified approach, realized in a system prototype QUALM, to seamlessly integrate TABDUAL and EVOLP/R by joint tabling of abduction and updating, in order to keep the benefit of tabling in each individual approach. Additionally, QUALM implements our LP-based counterfactual evaluation procedure that requires the interplay between abduction and updating, as described in Chapter 6.

7.3.1 Joint Tabling of Abduction and Updating

Our initial attempt, in Saptawijaya and Pereira (2014), depends on the notion of expectation, which is also employed in ACORDA and EPA, to express preconditions for enabling the assumption of an abducible. This concept consequently requires every rule with abducibles in its body to be preprocessed, by substituting abducible A with $consider(A)$.

While this notion of expectation is useful in constraining abducibles to those relevant for the problem at hand, its application in Saptawijaya and Pereira (2014) limits the benefit of tabling abductive solutions. That is, instead of associating tabling abductive solutions to an atom, say q (via a tabled predicate q_{ab}), a single and generic tabled predicate $consider_{ab}(A)$ is introduced for tabling just this one single given abducible A under consideration. Even though $expect(A)$ and $expect\_not(A)$ may have rules stating conditions for expecting A or otherwise, it is unnatural that such rules would have other abducibles as a condition for abducing A, as $consider/1$ concerns conditions for *one* abducible only, according to its intended semantics and use. That is, a $consider/1$ call must not depend on another $consider/1$, and so will never have an abductive context. Such a generic tabling by $consider_{ab}$ is superfluous and not intended by the concept of tabling abductive solutions introduced in TABDUAL. In this section we remedy the joint tabling approach of Saptawijaya and Pereira (2014) in order to uphold the idea and the benefit of tabling abductive solutions in contextual abduction.

The joint tabling approach in combining the techniques from TABDUAL and EVOLP/R naturally depends on two pieces of information carried from each of these techniques, viz. abductive contexts and the timestamp (i.e., holds-time) indicating when a fluent holds

true, respectively. They keep the same purpose in the integration as in their respective individual approach. Example 22 shows how both entries figure in a remedied rule transform.

Example 22 *Recall rule $s \leftarrow q, b$ in Example 3 of Chapter 5. Its rule transform below illustrates the joint tabling approach by combining the abductive contexts and the holds-time information:*

$$\begin{aligned} s_{ab}(E_3, H) \quad \leftarrow \quad & \#r(s,[q,b],[\,],E_1,H_r),\ b(E_1,E_2,H_b),\ q(E_2,E_3,H_q), \\ & latest([(\#r(s,[q,b]),H_r),(b,H_b),(q,H_q)],H). \end{aligned}$$

where $s_{ab}/2$ is now an incremental *tabled predicate, which later can be reused in the definition of $s/3$ (see rule 7.2 below).*

There are three important points in this transformation to clarify.

- Unlike the approach in Saptawijaya and Pereira (2014), the abducible b in the body is not required to be preprocessed into $consider(b)$. Furthermore, the abducible b is now called explicitly, rather than immediately placed as the input abductive context of q as in the TABDUAL transformation (cf. rule 5.3 in Chapter 5).

  This explicit call is intended to anticipate possible updates on this abducible. Such abducible updates may take place when one wants to commit to a preferred explanation and fix it in the program as a fact. For example, this is the case when an abduced background context of a counterfactual needs to be fixed, as shown in our counterfactual evaluation procedure in Chapter 6. Having an abducible as an explicit goal in the body thus facilitates bottom-up propagation of its updates, which is induced by incremental tabling of fluents, due to the dependency of the tabled predicate $s_{ab}/2$ in the head on the abducible goal in the body ($b/3$). This explicit call of abducible is resolved by the transformation of abducibles, similar to that of TABDUAL, cf. Definition 19 in Chapter 5, e.g., the transform rule for abducible b is as follows:

  $$b(I,O,H) \leftarrow insert\_abducible(b,I,O,H)$$

- Like in EVOLP/R, the rule name fluent $\#r(s,[q,b])$ is assigned to uniquely identify the rule $s \leftarrow q, b$, which now also has additional parameters of input and output abductive contexts besides its usual timestamp parameter H_r. Like in TABDUAL, E_3 is an abductive solution tabled via predicate s_{ab}, and obtained from relaying the ongoing abductive solution in context E_1 from the goal $\#r(s,[q,b])$ to the goal q in the body, given the empty input abductive context $[\,]$ of $\#r(s,[q,b])$. This empty input abductive context is due to the treatment of the abducible b, which now becomes an explicit goal in the body rather than appear in an input abductive context (as explained above).

  The reserved predicate $latest/2$ is borrowed from its EVOLP/R part, which determines the holds-time H of s_{ab} from the three goals in its body that latest holds. Note that if H_b is still a variable then it is set to the latest H.

- By prioritizing abducible goals to occur before any non-abducible goals in the body, the benefit of TABDUAL, viz., of reusing tabled abductive solutions from one context to another, can still be obtained. In Example 22, calling the abducible b, before q, with the empty input abductive context, provides q with an actual input abductive context $E_2 = [b]$. It thus achieves the same effect as simply having $[b]$ as the input context of q (cf. rule 5.3 in Chapter 5). Note that the rule name fluent $\#r(s, [q, b])$ is a fact, which transforms into $\#r(s, [q, b], I, I, 1)$, and therefore the empty context is just relayed intact to E_1.

The tabled solution in s_{ab} can be reused via the definition s below, which is similar to rule 5.4 in Chapter 5, except for the addition of timestamp information T:

$$s(I, O, T) \leftarrow s_{ab}(E, T), produce\_context(O, I, E). \tag{7.2}$$

where $produce\_context/3$ is defined as in TABDUAL.

Finally, the different purposes of the dual program transformation, employed both in TABDUAL and EVOLP/R, are consolidated within this unified approach. The abductive contexts and the timestamp information also jointly figure in the parameters of dual predicates, as shown in Example 23 below.

Example 23 *Recall Example 6 in Chapter 5. Considering $p/0$ as a fluent, the dual program transformation of the unified approach results in rules below. Note that each rule of $p/0$ is assigned a unique rule name.*

$$\begin{aligned}
not_p(T_0, T_2, H_p) &\leftarrow p^{*1}(T_0, T_1, D_{p_1}, H_{p_1}),\ p^{*2}(T_1, T_2, D_{p_2}, H_{p_2}),\\
&\qquad latest([(D_{p_1}, H_{p_1}), (D_{p_2}, H_{p_2})], H_p) \qquad (7.3)\\
p^{*1}(I, O, not_\#r(p, [a]), H_{p_{11}}) &\leftarrow not_\#r(p, [a], I, O, H_{p_{11}}).\\
p^{*1}(I, O, a^*, H_{p_{12}}) &\leftarrow a^*(I, O, H_{p_{12}}).\\
p^{*2}(I, O, not_\#r(p, [q, not\ r]), H_{p_{21}}) &\leftarrow not_\#r(p, [q, not\ r], I, O, H_{p_{21}}).\\
p^{*2}(I, O, not_q, H_{p_{22}}) &\leftarrow not_q(I, O, H_{p_{22}}).\\
p^{*2}(I, O, r, H_{p_{23}}) &\leftarrow r(I, O, H_{p_{23}}).
\end{aligned}$$

In each second layer dual rule p^{*i}, the chosen dualized negated literal needs to be passed to the first layer dual rule, in its parameter D_{p_i}. Like in EVOLP/R, *this information is needed by $latest/2$ to ensure that none of negated dualized literals in the body were subsequently supervened by their complements at some time before H_p. Indeed, passing this information achieves the same effect as if the dual rules are represented in a flattened form, i.e., if the goals p^{*1} and p^{*2} in the body of not_p are substituted by their chosen dualized negated literals.*

7.3.2 Evaluating Counterfactuals

For the purpose of evaluating counterfactuals, QUALM provides the construct $intervened(L)$ to declare all predicates, in list L, that are subject to intervention. For example, we have

$intervened([lightning/0])$ for the only counterfactual in Example 19 of Chapter 6. This declaration is useful to determine which rules to transform, according to the step 2 of our counterfactuals evaluation procedure (see Procedure 2 in Chapter 6). The transformation stage in step 2 (but not the intervention) can therefore be performed in advance, as a one-time transformation.

In QUALM, the state transition of the program, as a consequence of program updating (by asserting or retracting fluents, in our case), is facilitated by timestamps that are internally managed. Following the convention of EVOLP/R, the program is initially inserted at state (timestamp) $T = 1$, which subsequently progresses to $T = 2$ as the current state.

Starting in step 1 of Procedure 2, a top-level query $query(Query, In, Out)$ is invoked for finding the explanation, in the output abductive context Out, of the observation $Query$, given the input abductive context In.[2] For example:

$$?- \ query((lightning, fire, leavesOnGround), [\,], E)$$

provides an explanation E (for an empty input abductive context) to the observation $O = \{lightning, fire, leavesOnGround\}$ of Example 19. In order to fix $E = \{storm, barbecue^*\}$ as the abduced background context in evaluating counterfactual at the present state $T = 2$, both fluents *storm* and $barbecue^*$, that were true at the factual state $T_E = 1$, are asserted. QUALM provides a reserved predicate $updates(L)$ to record pending incremental assertion of fluents (and their assertion timestamps) in the list L. As in EVOLP/R, only those pending assertions whose timestamps up to some query-time will become actual, and this is triggered by a top-level query with its specific query-time.

Providing the transformation stage in step 2 has been performed in advance, the causal intervention "there had not been lightning" is enacted by the hypothetical update of fluent $make\_not(lightning)$, via $updates([make\_not(lightning)])$. As described in Section 6.2, this update strictly takes place between two consecutive factual states; in this case between $T_E = 1$ and the current state $T = 2$. QUALM internally assigns a fraction of timestamp, say 0.01, just after T_E, viz., the hypothetical update $make\_not(lightning)$ is imposed at state $T_H = 1.01$. It thus simulates an intervention via an update in the past, while keeping the current state at $T = 2$.

After this update, the validity of the present counterfactual (at $T = 2$) can be checked by testing its conclusion (step 3 of the procedure). For example, the top-level query $?- \ query(fire, [\,], E)$ is launched to ascertain whether forest fire would have occurred after the hypothetical update. QUALM answers 'no', which verifies the counterfactual's validity that the forest fire would not have occurred.

Finally, to reinstate the current factual situation from a counterfactual mode, the previous hypothetical update can be canceled by updating the program with its fluent complement. Continuing the above example, $updates([not\ make\_not(lightning)])$ is given,

[2]Predicate $query/3$ does not explicitly specify a parameter for query-time. In this case, the query-time always refers to the current timestamp. Alternatively, QUALM also provides $query(Query, In, Out, QTime)$, which allows specifying a particular query-time $QTime$.

and QUALM will internally assign a fraction of time after T_H for the timestamp T_F of this update, e.g., at $T_F = T_H + 0.01 = 1.02$. It thus supervenes the hypothetical update $make\_not(lightning)$ that was enacted at $T_H = 1.01$, and consequently, the intervention is no longer imposed on the program.

By using subsequent fractional timestamps to the first counterfactual, other counterfactuals may be queried assuming the hypothetical context of the previous ones, until the whole hypothetical situation is supervened by the above mechanism.

7.4 Concluding Remarks

As we show in the subsequent chapter, the different combinations of features in these three systems allows us to focus on particular morality issues to model. Note that while the three systems discussed here afford morality experiments, their applications clearly are not specific to morality.

The use of the XASP package in ACORDA and PROBABILISTIC EPA is important, as it shows that stable models are computed with respect to a residual program, the latter being itself obtained via a top-down query-oriented procedure (which maintains the relevancy property in finding an answer to a query). While PROBABILISTIC EPA is based on P-log for its probabilistic reasoning, it would be interesting in future to consider alternatives. In particular, since PROBABILISTIC EPA is implemented in XSB Prolog, the *Probabilistic Inference with Tabling and Answer subsumption (PITA)* (Riguzzi and Swift, 2011) package in XSB Prolog may be a good alternative candidate. In fact, PITA does not only support probabilistic logic programs, but may also be suitable for possibilistic logic programs. Such available options may benefit PROBABILISTIC EPA for its more general purpose applications.

QUALM is an ongoing work and continuously being improved. It is tempting to add features existing in the other two systems, e.g., preferences or probabilistic reasoning, into QUALM, in order to have a fully integrated system. But this should be prudently considered, as it will obviously increase the complexity of the system. We touch upon this general issue in Chapter 9.

CHAPTER 8

MODELING MORALITY USING LOGIC PROGRAMMING

This chapter aims at realizing our conception about representing diverse moral facets in Logic Programming, by modeling several issues pertaining to those moral facets, using the three systems discussed in Chapter 7. The applicability of these systems corresponds with their relevance to the moral issues being modeled.

In Section 8.1, ACORDA is employed to model moral permissibility, emphasizing the use of integrity constraints in abduction and preferences over abductive scenarios, where several cases of the classic trolley problem are modeled. Then, moral reasoning concerning uncertain actions is modeled, in Section 8.2, by means of PROBABILISTIC EPA. Finally, we demonstrate the use of QUALM for modeling the issue of moral updating and counterfactual moral reasoning, in Section 8.3.

8.1 Moral Reasoning with ACORDA

In Chapter 3, several cases built from the classic trolley problem (Foot, 1967) are presented. The cases concern themselves with the question of moral permissibility in a type of dilemma that involves harm. They are apparently similar in the dilemma they introduce (five people vs. one person being killed), yet the nuances in their specific scenarios and applied moral principles may influence moral permissibility judgments.

We model each case of the trolley problem in ACORDA separately, and demonstrate how appropriate moral decisions are delivered through abductive reasoning. By appropriate moral decisions we mean the ones that conform with those the majority of people make, based on empirical results in the literature. We particularly refer to Hauser (2007), Mikhail (2007), and Hauser et al. (2007), where experiments are conducted to assess moral

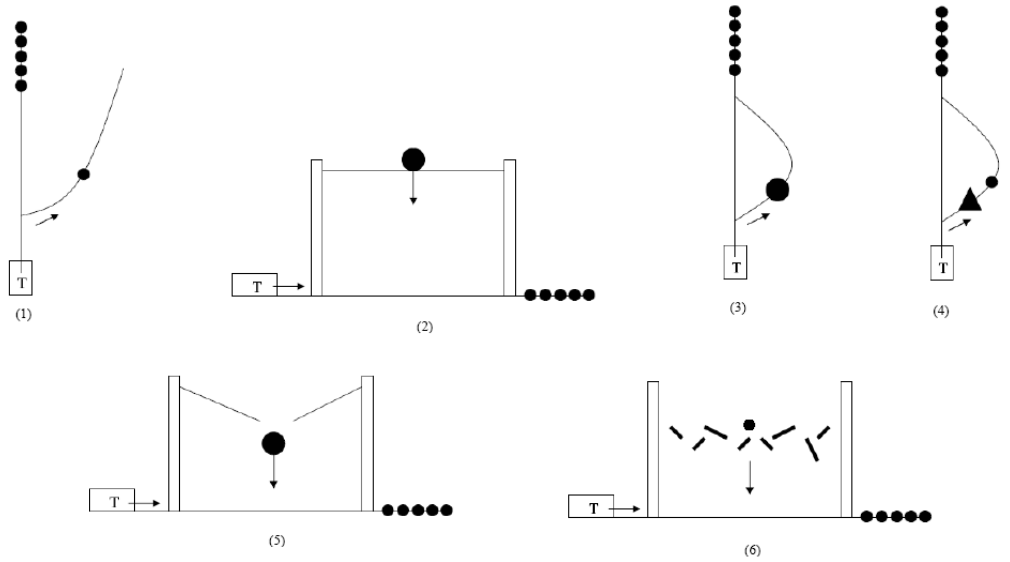


Figure 8.1: The six trolley cases: (1) Bystander, (2) Footbridge, (3) Loop, (4) Man-in-front, (5) Drop Man, (6) Collapse Bridge

judgments of subjects from demographically diverse populations (across gender, ethnicity, religion, age, exposure to moral coursework, etc.) on the following various trolley problem cases. The experiments are specifically intended to support their framework of human moral cognition, known as universal moral grammar (Hauser, 2007; Mikhail, 2007), analogously to Chomsky's universal grammar in language. It provides universal moral principles, albeit being culturally adjustable, that enable an individual to evaluate what actions are permissible, obligatory, or forbidden.

We first describe various cases of the trolley problem. All cases present a moral dilemma that inquires whether it is permissible to harm an individual for the purpose of saving others. The initial circumstances are the same (Hauser, 2007):

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

Given the above initial circumstance, we consider below six cases of moral dilemmas (Mikhail, 2007). These six cases are described below and visually depicted in Figure 8.1.

1. **Bystander.** Hank is standing next to a switch, which he can throw, that will turn the trolley onto a parallel side track, thereby preventing it from killing the five people. However, there is a man standing on the side track with his back turned. Hank can throw the switch, killing him; or he can refrain from doing this, letting the five die. Is it morally permissible for Hank to throw the switch?

2. **Footbridge.** Ian is on the footbridge over the trolley track. He is next to a heavy

man, which he can shove onto the track in the path of the trolley to stop it, thereby preventing it from killing the five people. Ian can shove the man onto the track, resulting in death; or he can refrain from doing this, letting the five die. Is it morally permissible for Ian to shove the man?

3. **Loop.** Ned is standing next to a switch, which he can throw, that will temporarily turn the trolley onto a loop side track. There is a heavy object on the side track. If the trolley hits the object, the object will slow the train down, giving the five people time to escape. The heavy object is a man. Ned can throw the switch, preventing the trolley from killing the five people, but killing the man. Or he can refrain from doing this, letting the five die. Is it morally permissible for Ned to throw the switch?

4. **Man-in-front.** Oscar is standing next to a switch, which he can throw, that will temporarily turn the trolley onto a looping side track. There is a heavy object on the side track. If the trolley hits the object, the object will slow the train down, giving the five people time to escape. There is a man standing on the side track in front of the heavy object. Oscar can throw the switch, preventing the trolley from killing the five people, but killing the man. Or he can refrain from doing this, letting the five die. Is it morally permissible for Oscar to throw the switch?

5. **Drop Man.** Victor is standing next to a switch, which he can throw, that will drop a heavy object into the path of the trolley, thereby stopping the trolley and preventing it from killing the five people. The heavy object is a man, who is standing on a footbridge overlooking the track. Victor can throw the switch, killing him; or he can refrain from doing this, letting the five die. Is it morally permissible for Victor to throw the switch?

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

Interestingly, despite the same dilemma, viz., to save five albeit killing one, subjects come to different judgments on whether the action to reach the goal is permissible or impermissible, as summarized in Table 8.1 (by taking the majority from the result of each case as presented in Mikhail (2007)). Even though the subjects are unable to explain the moral principles in their attempts at justification, their moral judgments are consistent with the Doctrine of Double Effect (McIntyre, 2004). This moral principle addresses the permissibility of an action that causes a harm by distinguishing whether this harm is a mere *side-effect* of bringing about a good result (in which case it is permissible), or rather an *intended means* to bringing about the same good end (in which case it is impermissible).

Table 8.1: Summary of moral judgments for the six trolley problem cases

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

The Doctrine of Double Effect is modeled subsequently via integrity constraints and a posteriori preferences to capture the deontological and the utilitarian flavours of moral judgments, respectively, in these cases. Possible decisions are modeled as abducibles. Moral decisions are therefore made by first satisfying integrity constraints (to rule out impermissible actions), then computing abductive stable models from the resulting abductive scenarios, and finally preferring amongst them (by means of LP rules) on the basis of their consequences of abducibles in the models.

In addition to the Doctrine of Double Effect, the Doctrine of Triple Effect (Kamm, 2006) is also considered, which extends our LP representation of the Loop Case – the case affected by this doctrine – and contrasts moral permissibility of this specific case with respect to the Doctrine of Double Effect.

Next, we detail the modeling of the above six cases in ACORDA. In each case of the trolley problem, there are always two possible decisions to make. One of these is the same for all cases, i.e. letting the five people die by merely watching the train go straight. The other decision depends on the cases, e.g. throwing the switch, shoving a heavy man, or the combination of them. In order to assess how flexible is our model of the moral rule, we additionally model other variants for the Footbridge and Loop cases.

Modeling the Bystander Case

This case is modeled as follows:

$$
\begin{array}{lcl}
side\_track. & & on\_side(john). \qquad human(john). \\
expect(watch). & & \\
train\_straight & \leftarrow & consider(watch). \\
end(die(5)) & \leftarrow & train\_straight. \\
observed\_end & \leftarrow & end(X). \\
expect(throw\_switch) & \leftarrow & side\_track. \\
turn\_side & \leftarrow & consider(throw\_switch). \\
kill(1) & \leftarrow & human(X), on\_side(X), turn\_side. \\
end(save\_men, ni\_kill(N)) & \leftarrow & turn\_side, kill(N). \\
observed\_end & \leftarrow & end(X, Y).
\end{array}
$$

The first three facts describe that there is a side track and a man (here, named *john*) standing on that track. The fact $expect(watch)$ and rule $expect(throw\_switch) \leftarrow side\_track$ indicate that watching and throwing the switch, respectively, are two available abducibles, representing possible decisions Hank has. In this case, the action of throwing switch is only expected as an abducible, if the side track exists. The other clauses represent the chain of actions and consequences for these two abducibles.

The predicate $end(die(5))$ represents the final consequence of abducing *watch*, that five people die. On the other hand, the predicate $end(save\_men, ni\_kill(N))$ represents the final consequence of abducing *throw_switch*: it will save the five without intentionally killing someone. The way of representing these two consequences is chosen differently, because these two abducibles are of different kind. Merely watching the trolley go straight is an omission of action that just has negative consequence, whereas throwing the switch is an action that is performed to achieve a goal and additionally has negative consequence. Since abducibles in other cases of the trolley problem also share this property, this way of representing will be used in other cases too. Finally, *observed_end* is used for encapsulating both representations into one, to serve as a goal that will later be triggered by an integrity constraint.

Modeling the Footbridge Case

The program below models the action of shoving an object as an abducible, together with its chain of consequences. The part for the decision of merely watching is the same as in the case of Bystander.

$$
\begin{array}{lcl}
stand\_near(john). & & human(john). \qquad heavy(john). \\
expect(shove(X)) & \leftarrow & stand\_near(X). \\
on\_track(X) & \leftarrow & consider(shove(X)). \\
stop\_train(X) & \leftarrow & on\_track(X), heavy(X). \\
kill(1) & \leftarrow & human(X), on\_track(X). \\
kill(0) & \leftarrow & inanimate\_object(X), on\_track(X). \\
end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), stop\_train(X), kill(N). \\
end(save\_men, i\_kill(N)) & \leftarrow & human(X), stop\_train(X), kill(N). \\
observed\_end & \leftarrow & end(X, Y).
\end{array}
$$

The fact of a heavy man (here, also named *john*) on the footbridge standing near to the agent is modeled similarly as in the Bystander case. This case can be made more interesting by additionally having another inanimate heavy object (e.g., rock) on the footbridge near to Ian, which corresponds to the first variant of the Footbridge case:

$$stand\_near(rock). \qquad inanimate\_object(rock). \qquad heavy(rock).$$

Alternatively, in the second variant, an even loop over default negation can be used, to represent that either a man or an inanimate object be on the footbridge next to Ian:

$$stand\_near(john) \leftarrow not\ stand\_near(rock).$$
$$stand\_near(rock) \leftarrow not\ stand\_near(john).$$

Note that the action of shoving an object is only possible if there is an object near Ian to shove, hence the rule $expect(shove(X)) \leftarrow stand\_near(X)$. We also have two clauses that describe two possible final consequences. The rule with the head $end(save\_men, ni\_kill(N))$ deals with the consequence of reaching the goal, viz., saving the five, but not intentionally killing someone (in particular, without killing anyone in this case). On the other hand, the rule with the head $end(save\_men, i\_kill(N))$ expresses the consequence of reaching the goal but involving an intentional killing.

Modeling the Loop Case

We consider three variants for the loop track case. In the first variant, instead of having only one looping side track as in the original scenario, we consider two looping side tracks, viz., the left and the right loop side tracks. John, a heavy man, is standing on the left side track, whereas on the right side track there is an inanimate heavy object, e.g., rock. These facts can be represented as follows:

$$\begin{array}{lll} side\_track(left). & side\_track(right). & \\ on(john, left). & human(john). & heavy(john). \\ on(rock, right). & inanimate\_object(rock). & heavy(rock). \end{array}$$

The switch can be thrown to either one of the two looping side tracks, represented with the abducible predicate $throw\_switch/1$. The expectation of this abducible together with the chain of consequences are shown below:

$$\begin{array}{lcl} expect(throw\_switch(Z)) & \leftarrow & side\_track(Z). \\ turn\_side(Z) & \leftarrow & consider(throw\_switch(Z)). \\ slowdown\_train(X) & \leftarrow & turn\_side(Z), on(X, Z), heavy(X). \\ kill(1) & \leftarrow & turn\_side(Z), on(X, Z), human(X). \\ kill(0) & \leftarrow & turn\_side(Z), on(X, Z), inanimate\_object(X). \\ end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), slowdown\_train(X), kill(N). \\ end(save\_men, i\_kill(N)) & \leftarrow & human(X), slowdown\_train(X), kill(N). \\ observed\_end & \leftarrow & end(X, Y). \end{array}$$

In the second variant, we consider a single looping side track with either a man or an inanimate object on it. As in the footbridge case, rather than having two separate programs to model these distinct facts, we can model it instead using an even loop over default negation to capture alternatives between objects:

$$side\_track(john) \leftarrow not\ side\_track(rock).$$
$$side\_track(rock) \leftarrow not\ side\_track(john).$$

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

$$
\begin{array}{lcl}
expect(throw\_switch(X)) & \leftarrow & side\_track(X). \\
turn\_side(X) & \leftarrow & consider(flipping\_switch(X)). \\
slowdown\_train(X) & \leftarrow & turn\_side(X), heavy(X). \\
kill(1) & \leftarrow & turn\_side(X), human(X). \\
kill(0) & \leftarrow & turn\_side(X), inanimate\_object(X). \\
end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), slowdown\_train(X), kill(N). \\
end(save\_men, i\_kill(N)) & \leftarrow & human(X), slowdown\_train(X), kill(N). \\
observed\_end & \leftarrow & end(X, Y).
\end{array}
$$

Finally, the third variant concerns the distinction between the Doctrines of Double Effect and Triple Effect. Recall from Chapter 3 that the Doctrine of Triple Effect refines the Doctrine of Double Effect particularly on the notion about harming someone as an intended means. That is, the Doctrine of Triple Effect distinguishes further between doing an action *in order* that harm occurs and doing it *because* harm will occur. Like in the Doctrine of Double Effect, the former is impermissible, whereas the latter is permissible. The third variant thus corresponds to the *Loop-Push Case* discussed in Section 3.1 of Chapter 3, which is revisited here.

In this variant, the looping side track is initially empty, but there is a footbridge over the side track with a heavy man on it. Instead of only diverting the trolley onto the empty looping side track, an ancillary action of shoving a man onto the looping side track is performed. That is, the action of throwing the switch, for diverting the trolley, is followed by the action of shoving a man, to place the man onto the empty looping side track. As in the original Loop case, there is only a single side track, which we can represent by using a simple fact *side_track*.

In order to model both the original Loop case and the Loop-Push case in one program, we can again use an even loop over default negation to represent whether a man has already been standing on the looping side track or the looping side track is initially empty.

$$
\begin{array}{c}
man\_sidetrack \leftarrow not\ empty\_sidetrack. \\
empty\_sidetrack \leftarrow not\ man\_sidetrack.
\end{array}
$$

We have three actions available, represented as abducibles, with the chain of consequences modeled in the program below. In this model, the abducible *watch* and its consequence are similar as in other previous cases. The action *throw_switch* is expected, if there is a side track controlled by the switch. Throwing the switch will turn the trolley to the side track. If the man has already been standing on the side track and the trolley also turns to the side track, then the five people are saved, but also killing the man standing on the side track. Finally, the action of *shoving* is expected, if the side track is empty and

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

$$
\begin{array}{lcl}
expect(watch). & & \\
train\_straight & \leftarrow & consider(watch). \\
end(die(5)) & \leftarrow & train\_straight. \\
observed\_end & \leftarrow & end(X). \\
 & & \\
expect(throw\_switch) & \leftarrow & side\_track. \\
turn\_side & \leftarrow & consider(throw\_switch). \\
end(save\_men, standing\_hitting) & \leftarrow & man\_sidetrack, turn\_side. \\
 & & \\
expect(shoving) & \leftarrow & empty\_sidetrack, consider(throw\_switch). \\
freely\_goto\_maintrack & \leftarrow & empty\_sidetrack, turn\_side, not\ consider(shoving). \\
end(die(5)) & \leftarrow & freely\_goto\_maintrack. \\
place\_man\_sidetrack & \leftarrow & consider(shoving). \\
end(save\_men, placing\_hitting) & \leftarrow & place\_man\_sidetrack, turn\_side. \\
observed\_end & \leftarrow & end(X, Y).
\end{array}
$$

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

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

Modeling the Man-in-front Case

Recall that in the Man-in-front case there is a heavy (inanimate) object, e.g., rock, on the looping side track, onto which the trolley can turn. Additionally, there is a man standing in front of the heavy object. These facts can be modeled as:

$$
\begin{array}{lll}
side\_track. & & \\
on\_side(rock). & inanimate\_object(rock). & heavy(rock). \\
in\_front\_of(rock, john). & human(john). &
\end{array}
$$

The following rules model the throwing switch action as an abducible together with

its chain of consequences:

$$
\begin{array}{lcl}
expect(throw\_switch) & \leftarrow & side\_track. \\
turn\_side & \leftarrow & consider(throw\_switch). \\
kill(1) & \leftarrow & turn\_side, on\_side(X), in\_front\_of(X,Y), human(Y). \\
slowdown\_train(X) & \leftarrow & turn\_side, on\_side(X), heavy(X). \\
end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), slowdown\_train(X), kill(N). \\
end(save\_men, i\_kill(N)) & \leftarrow & human(X), slowdown\_train(X), kill(N). \\
observed\_end & \leftarrow & end(X,Y).
\end{array}
$$

Modeling the Drop Man Case

This case is modeled as:

$$
\begin{array}{ll}
switch\_connected(bridge). & on(bridge, john). \\
human(john). & heavy(john).
\end{array}
$$

$$
\begin{array}{lcl}
expect(throw\_switch(Z)) & \leftarrow & switch\_connected(Z). \\
drop(X) & \leftarrow & consider(throw\_switch(Z)), on(Z,X). \\
kill(1) & \leftarrow & human(X), drop(X). \\
stop\_train(X) & \leftarrow & heavy(X), drop(X). \\
end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), stop\_train(X), kill(N). \\
end(save\_men, i\_kill(N)) & \leftarrow & human(X), stop\_train(X), kill(N). \\
observed\_end & \leftarrow & end(X,Y).
\end{array}
$$

In the above model, the fact that the switch is connected to the bridge is the condition for the action of throwing the switch to be available.

Modeling the Collapse Bridge Case

The Collapse Bridge case is a variation from the Drop Man case. In this case, the footbridge itself is the heavy object which may stop the trolley when it collapses and prevents the train from killing the five people. There is also a man, John, standing on the footbridge, as in the Drop Man case.

The model of this case is just a slight modification from that of the Drop Man case:

$switch\_connected(bridge).$ $heavy(bridge).$ $inanimate\_object(bridge).$
$human(john).$ $on(bridge, john).$

$$
\begin{array}{lcl}
expect(throw\_switch(X)) & \leftarrow & switch\_connected(X). \\
collapse(X) & \leftarrow & consider(throw\_switch(X)). \\
stop\_train(X) & \leftarrow & heavy(X), collapse(X). \\
kill(1) & \leftarrow & human(Y), on(X,Y), collapse(X). \\
end(save\_men, ni\_kill(N)) & \leftarrow & inanimate\_object(X), stop\_train(X), kill(N). \\
end(save\_men, i\_kill(N)) & \leftarrow & human(X), stop\_train(X), kill(N). \\
observed\_end & \leftarrow & end(X,Y).
\end{array}
$$

The next two sections detail how the Doctrine of Double Effect (and similarly, the Doctrine of Triple Effect) is modeled by a combination of a priori integrity constraints and a posteriori preferences.

8.1.1 Deontological Judgments via a Priori Integrity Constraints

In this application, integrity constraints are used for two purposes.

First, they are utilized to force the goal in each case, by observing the desired end goal resulting from each possible decision:

$$false \leftarrow not\ observed\_end.$$

Such an integrity constraint thus enforces all available decisions to be abduced, together with their consequences, from all possible observable hypothetical end goals in the problem representation.

The second purpose of integrity constraints is for ruling out impermissible actions, viz., actions that involve intentional killing in the process of reaching the goal. This can be enforced by the integrity constraint:

$$false \leftarrow intentional\_killing. \tag{8.1}$$

The definition of *intentional_killing* depends on rules in each case considered and whether the Doctrines of Double Effect or Triple Effect is to be upheld. With respect to our model, for the Doctrine of Double Effect, it is defined as:

$$intentional\_killing \leftarrow end(save\_men, i\_kill(Y)).$$

whereas for the Doctrine of Triple Effect:

$$intentional\_killing \leftarrow end(save\_men, placing\_hitting).$$

The integrity constraint 8.1 above serve as the first filter of abductive stable models, by ruling out impermissible actions. This is in line with deontological viewpoint of a moral

judgment, which should conform to some moral principle, regardless of how good its consequence is. In this application, regardless of how good the consequence of a harming action is, if this action is performed as an intended means to a greater good (e.g., shoving the heavy man), then according to the Doctrine of Double Effect (and the Doctrine of Triple Effect accordingly) this action is impermissible, and hence, it is ruled out by the integrity constraint. In other words, the integrity constraint eventually affords us with just those abductive stable models (computed by means of the XASP package) that contain only permissible actions.

8.1.2 Utilitarian Judgments via a Posteriori Preferences

One can further prefer amongst the permissible actions those resulting in greater good. That is, if an a priori integrity constraint corresponds to the agent's fast and immediate response, generating intended decisions that comply with deontological ethics (achieved by ruling out the use of intentional harm), then a posteriori preferences amongst permissible actions correspond to a slower response, as they involve a more involved reasoning on action-generated models in order to capture utilitarianism (which favors welfare-maximizing behaviors), cf. the psychological empirical tests reported by Cushman et al. (2010); Greene et al. (2004) in the dual-process model (Section 3.2).

In this application, a preference predicate is generally defined to select those abductive stable models containing decisions with greater good of consequences. The following rules of *select*/2 achieves this purpose:

$$select(Xs, Ys) \leftarrow select(Xs, Xs, Ys).$$

The first argument of this predicate refers to the set of initial abductive stable models (obtained after applying a priori integrity constraints) to prefer, whereas the second argument refers to the preferred ones. The auxiliary predicate *select*/3 only keeps abductive stable models that contain decisions with greater good of consequences. In the trolley problem cases, the greater good is evaluated by a utility function concerning the number of people that die as a result of possible decisions. This is realized in the definition of predicate *select*/3 by comparing final consequences that appear in the initial abductive stable models.

$$\begin{array}{lcl}
select([\,], _, [\,]). & & \\
select([X|Xs], Zs, Ys) & \leftarrow & member(end(die(N)), X), \\
 & & member(Z, Zs),\ member(end(save_men, ni_kill(K)), Z), \\
 & & N > K,\ select(Xs, Zs, Ys). \\
select([X|Xs], Zs, Ys) & \leftarrow & member(end(save_men,\ ni_kill(K)), X), \\
 & & member(Z, Zs),\ member(end(die(N)), Z), \\
 & & N =< K,\ select(Xs, Zs, Ys). \\
select([X|Xs], Zs, [X|Ys]) & \leftarrow & select(Xs, Zs, Ys).
\end{array}$$

The first clause of *select*/3 is the base case. The second clause and the third clause together eliminate abductive stable models containing decisions with worse consequences, whereas

Table 8.2: Summary of preferred abductive stable models in the six trolley cases

Case	Initial Models	Final Models
Bystander	$[throw\_switch], [watch]$	$[throw\_switch]$
Footbridge(1)	$[watch], [shove(rock)]$	$[shove(rock)]$
Footbridge(2)	$[watch, stand\_near(john)]$, $[watch, stand\_near(rock)]$, $[shove(rock)]$	$[watch, stand\_near(john)]$, $[shove(rock)]$
Loop(1)	$[throw\_switch(right)]$, $[watch]$	$[throw\_switch(right)]$
Loop(2)	$[watch, side\_track(john)]$, $[watch, side\_track(rock)]$, $[throw\_switch(rock)]$	$[watch, side\_track(john)]$, $[throw\_switch(rock)]$
Loop(3)	$[watch, empty\_sidetrack]$, $[watch, man\_sidetrack]$, $[throw\_switch, empty\_sidetrack]$, $[throw\_switch, man\_sidetrack]$	$[watch, empty\_sidetrack]$, $[throw\_switch, man\_sidetrack]$
Man-in-front	$[watch]$, $[throw\_switch(rock)]$	$[throw\_switch(rock)]$
Drop Man	$[watch]$	$[watch]$
Collapse Bridge	$[watch]$, $[throw\_switch(bridge)]$	$[throw\_switch(bridge)]$

the fourth clause will keep those models that contain decisions with greater good of consequences.

Besides this quantitative measure, a specific qualitative preference rule can be applied to the second variant of the Footbridge case, where either a man or an inanimate object is on the footbridge next to Ian. Recall that this exclusive alternative is specified by an even loop over default negation, where we have an abductive stable model containing the consequence of letting die the five people even when a rock next to Ian. This model is certainly not the one we would like our moral reasoner to prefer. The following qualitative preference rules achieves this purpose:

$$
\begin{array}{lll}
select([\,],[\,]). & & \\
select([X|Xs],Ys) & \leftarrow & member(end(die(N)),X),\ member(stand\_near(rock),X),. \\
 & & select(Xs,Ys). \\
select([X|Xs],[X|Ys]) & \leftarrow & select(Xs,Ys).
\end{array}
$$

Table 8.2 gives a summary of preferred models in our experiments for all cases of the trolley problem. Column Initial Models contains abductive stable models obtained after applying a priori integrity constraints, but before a posteriori preferences, whereas column Final Models refers to those after a posteriori preferences are applied. Here, only relevant literals are shown.

Note that entry Footbridge(1) and Footbridge(2) refer to the first and second variants of the Footbridge case, respectively. Similarly, Loop(1), Loop(2), and Loop(3) refer to the first, second, and third variants of the Loop case, respectively. Both Loop(1) and Loop(2) employ the Doctrine of Double Effect. Consequently, there is no initial model (of these two cases) that contains the abducible about throwing the switch when a man is on a looping side track. This model has been ruled out by the integrity constraint 8.1, as it is considered impermissible according to the Doctrine of Double Effect. On the other hand, Loop(3), which employs the Doctrine of Triple Effect, does have an initial model with a man on the side track and throwing the switch as an abducible. This model is not ruled out by the integrity constraint 8.1, as it is deemed permissible by the Doctrine of Triple Effect.

The concept of inspection point (see Section 4.2) can be useful particularly to explain the moral reasoning behind the Loop(3) variant, where the Doctrine of Triple Effect is employed. In this variant, the fact that the man, an obstacle to the trolley, was already standing on the side track can be treated as an inspection point, which does not induce any additional abduction. In this case, the action of throwing the switch is the only action abduced to prevent the trolley from hitting the five people. On the other hand, in the empty sidetrack alternative, mere watching is not enough, as an extra abducible is required to shove the man onto the track in order to deliberately make him an obstacle where there was none, thus preventing the trolley from hitting the five people. However, the abductive stable model containing this further action has been ruled out by the integrity constraint 8.1. It thus conforms with the Doctrine of Triple Effect, as intentionally shoving the man, in addition to the act of throwing the switch, in order the trolley to hit the man for the purpose of stopping the trolley, is impermissible.

8.2 Moral Reasoning with PROBABILISTIC EPA

Moral reasoning is commonly 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 and certain information about the actions. It is therefore 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 sometimes required to proffer rulings beyond reasonable doubt. There is a vast body of research on proof beyond reasonable doubt within the legal community (see e.g., Newman (2006)).

The example contrived in the thesis is a variant of the Footbridge case, which is couched in court terms. It is by no means intended to express the full complexity found in courts. Nevertheless, it is sufficient to capture the deliberative employment of Scanlonian contractualism, where permissibility of actions – referring to the Doctrine of Double Effect – is addressed through justified but defeasible argumentative considerations. We consider a variant of the Footbridge case in Example 24 below.

Example 24 *Suppose a board of jurors 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 trolley track and the agent was seen on the bridge at the occasion. Is Ian guilty (beyond reasonable doubt), in the sense of violating the Doctrine of Double Effect, 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 Example 24.

Given the active goal *judge*, two abducibles are available: $verdict(guilty\_brd)$ and $verdict(not\_guilty)$, where *guilty_brd* stands for 'guilty beyond reasonable doubt'. Depending on how probable each possible verdict is, either the abducible $verdict(guilty\_brd)$ or $verdict(not\_guilty)$ is expected a priori.

$$abds([verdict/1]). \qquad on\_observe(judge).$$
$$judge \leftarrow verdict(guilty\_brd).$$
$$judge \leftarrow verdict(not\_guilty).$$
$$expect(verdict(X)) \leftarrow highly\_probable(X).$$

The probabilistic part is shown in the P-log(XSB) code below. The sort *intentionality* in line 1 represents the possibilities of an action being performed intentionally (*int*) or not (*not_int*). Random attributes df_run and *br_slip* in lines 2 and 3 denote two kinds of evidence: Ian was definitely running on the bridge in a hurry and the bridge was slippery at the time, respectively. Each has prior probability of $4/10$. The probability of intentional shoving is captured by the random attribute *shoved* (line 4), which is causally influenced by both evidences. Line 6 defines when the verdicts (*guilty* and *not_guilty*) are considered highly probable using the meta-probabilistic predicate $pr\_iShv/1$, shown by line 5. 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 $evd\_slip/1$).

```
beginPlog.
1. bool = {t, f}.   intentionality = {int, not_int}.
2. df_run : bool.   random(rdr,df_run,full).
   pa(rdr,df_run(t),d_(4, 10)).
3. br_slip : bool.  random(rsb,br_slip,full).
   pa(rsb,br_slip(t),d_(4, 10)).
4. 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).
```

```
5. 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).
6. highly_probable(guilty_brd) :- pr_iShv(PrG), PrG > 0.95.
   highly_probable(not_guilty) :- pr_iShv(PrG), PrG < 0.6.
endPlog.
```

Based on this representation, different judgments can be delivered by PROBABILISTIC EPA, subject to available (observed) evidences and their attending truth value. By viewing the standard probability of proof beyond reasonable doubt –here the value of 0.95 is adopted (Newman, 2006)– as a common ground for the probability of guilty verdicts to be qualified as 'beyond reasonable doubt', a form of argumentation may take place through presenting different evidence (via updating of observed evidence atoms, e.g., $evd\_run(true)$, $evd\_slip(false)$, etc.) as a consideration to justify an exception. Whether the newly available evidence is accepted as a justification to an exception –defeating the judgment based on the priorly presented evidence– depends on its influence on the probability $pr\_iShv(P)$ of intentional shoving, and thus eventually influences the final verdict. That is, it depends on whether this probability is still within the agreed standard of proof beyond reasonable doubt. We illustrate this with this scenarios below:

- If both evidences are available and true, i.e., it is known that the agent was running in a hurry ($evd\_run(true)$) on the slippery bridge $evd\_slip(true)$, then it may have bumped the man accidentally, shoving him unintentionally onto the track. This case obtains $pr\_iShv(0.05)$ in our model, allowing to abduce $verdict(not\_guilty)$ as the solution for *judge* (in this application, the threshold for *not_guilty* is 0.6).

- On the other hand, if the evidence later reveals that the agent was not running in a hurry ($evd\_run(false)$) and the bridge was not slippery ($evd\_slip(false)$), the model obtains $pr\_iShv(0.97)$, and it being greater than 0.95, $verdict(guilty\_brd)$ becomes the abductive solution of *judge*. Indeed, such evidences do not support the explanation that the man was shoved unintentionally (by accidental bumping): the action of shoving is more likely to have been performed intentionally, thus justifying now $verdict(guilty\_brd)$.

- Yet, if it is only known the bridge was not slippery ($evd\_slip(false)$) and no other evidence is available, then $pr\_iShv(0.80)$ and no abductive solution is returned (neither it is less than 0.6 nor greater than 0.95). This translates into the need for more evidence, as the available one is not enough to issue judgment.

In Han et al. (2012), PROBABILISTIC EPA (i.e., EPA with its P-log implementation) is also employed to extend the Bystander and the Footbridge cases, by introducing different aspects of uncertainty, such as the success in performing hypothesized actions (e.g., "how probable is it the five people will die?", "how probable the body of the shoved man will stop the trolley?") as well as beliefs and behaviors of other agents involved in the situation (e.g., "how probable the shoved man will take revenge, given that he may escape from being hit by the trolley?"). The revised Bystander and Footbridge cases are reported elsewhere (Han, 2009), so it is not repeated here.

8.3 Moral Reasoning with QUALM

In this section, two applications with QUALM are presented. The first application (Section 8.3.1) concerns moral updating, where the interplay between tabling in contextual abduction and updating for imposing a moral rule into effect are demonstrated. In the second application (Section 8.3.2), we explore the use of counterfactuals to address the issue of moral permissibility according to the Doctrines of Double Effect and Triple Effect, and to justify it.

8.3.1 Moral Updating

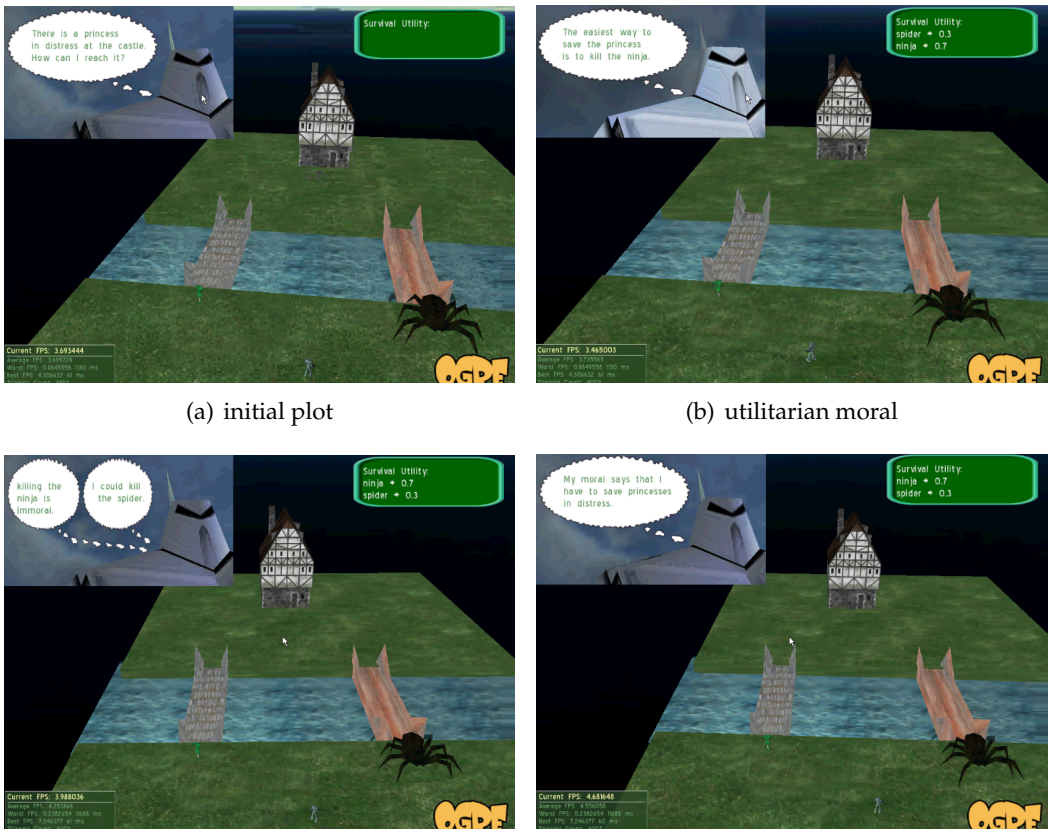
Moral updating (and evolution) concerns the adoption of new (possibly overriding) moral rules on top of those an agent currently follows. Such adoption often happens in the light of situations faced by the agent, e.g., when an authority contextually imposes other moral rules, or due to cultural difference.

This is not only relevant in a real world setting, but also in imaginary ones. In Lopes and Pereira (2010a), moral updating is illustrated in an interactive storytelling, and modeled using ACORDA. In this fantasy setting (Figure 8.2), a 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, it 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 (a giant spider or a human ninja) of the wizard which imprisoned the princess. In order to rescue the princess, the robot will have to defeat one of the minions to proceed. For a visual demo, see Lopes and Pereira (2010b).

This storytelling is reconstructed in the thesis using QUALM, to demonstrate in particular:

1. The direct use of LP updating so as to put an imposed moral rule into effect, via QUALM's rule name fluent mechanism to switch a rule on or off.

2. The relevance of contextual abduction to rule out abductive solutions, when a goal is invoked by a non-empty initial abductive context (the content of this context may be obtained from another agent, e.g., imposed by the princess).



(a) initial plot (b) utilitarian moral

(c) 'Gandhi' moral (d) 'knight' moral

Figure 8.2: Several plots of the interactive moral storytelling "Princess Savior Moral Robot"

A simplified program modeling the knowledge of the princess-savior robot in QUALM is shown below, where $fight/1$ is an abducible predicate:

$$
\begin{aligned}
guard(spider). \qquad & guard(ninja). \qquad human(ninja).\\
utilVal(spider, 0.3). \qquad & utilVal(ninja, 0.7).\\
survive\_from(G) &\leftarrow utilVal(G, V), V > 0.6.\\
intend\_savePrincess &\leftarrow guard(G), fight(G), survive\_from(G). \qquad (8.2)\\
intend\_savePrincess &\leftarrow guard(G), fight(G). \qquad (8.3)
\end{aligned}
$$

The first rule of *intend_savePrincess* (rule 8.2) corresponds to a utilitarian moral rule (with respect to the robot's survival), whereas the second one (rule 8.3) to a 'knight' moral, viz., to intend the goal of saving the princess at any risk (irrespective of the robot's survival chance). Since each rule in QUALM is assigned a unique name in its transform, the name of each rule for *intend_savePrincess* may serve as a unique moral rule identifier, say via rule name fluents $\#r(utilitarian)$ and $\#r(knight)$, respectively.[1] In the subsequent plots, unless

[1]The choice of these rule name fluents is for clarity of the presentation. They are more descriptive than that in the form of $\#r(H, [B])$ for rule $H \leftarrow B$.

otherwise stated, query

$$?-\ query(intend\_savePrincess, [\,], O)$$

is referred, representing the robot's intent on saving the princess (given an empty initial input abductive context).

- In the first plot, when both rule name fluents are retracted, the robot does not adopt any moral rule to save the princess, i.e., the robot has no intent to save the princess, and thus the princess is not saved.

- In the second plot, in order to maximize its survival chance in saving the princess, the robot updates itself with utilitarian moral, i.e., the program is updated with $\#rule(utilitarian)$. The robot thus abduces $O = [fight(ninja)]$ so as to successfully defeat the ninja instead of confronting the humongous spider.

- The use of tabling in contextual abduction is demonstrated in the third plot. Assuming that the truth of $survive\_from(G)$ implies the robot success in defeating (killing) guard G, the princess argues that the robot should not kill the human ninja, as it violates the moral rule she follows, say 'Gandhi' moral, expressed by the following rule in her knowledge (the first three facts in the robot's knowledge are shared with the princess):

  $$follow\_gandhi \leftarrow guard(G), human(G), fight^*(G).$$

  That is, from the query $?-\ query(follow\_gandhi, [\,], O_p)$, the princess abduces $O_p = [fight^*(ninja)]$, and imposes this abductive solution as the initial (input) abductive context of the robot's goal, i.e., via the query:

  $$?-\ query(intend\_savePrincess, [fight^*(ninja)], O_r).$$

  This input context is inconsistent with the tabled abductive solution $fight(ninja)$, and as a result, the query fails. In this case, the imposed 'Gandhi' moral conflicts with its utilitarian rule. In the visual demo (Lopes and Pereira, 2010b), the robot reacts by leaving its mission.

- In the final plot, as the princess is not saved, she further argues that she definitely has to be saved, by now additionally imposing on the robot the 'knight' moral. This amounts to updating the rule name fluent $\#rule(knight)$ so as to switch on the corresponding rule. As the goal *intend_savePrincess* is still invoked with the input abductive context $fight^*(ninja)$, the robot now abduces $O_r = [fight(spider)]$ in the presence of the newly adopted 'knight' moral. Unfortunately, it fails to survive, as confirmed by the failing of the query $?-\ survive\_from(spider)$.

The plots in this story reflect a form of deliberative employment of moral judgments within Scanlonian contractualism. For instance, in the second plot, the robot may justify its action to fight (and kill) the ninja due to the utilitarian moral it adopts. This justification is counter-argued by the princess in the subsequent plot, making an exception in saving her, by imposing the 'Gandhi' moral, disallowing the robot to kill a human guard. In this application, rather than employing updating, this exception is expressed via contextual abduction with tabling. The robot may justify its failing to save the princess (as the robot is leaving the scene) by arguing that the two moral rules it follows (viz., utilitarian and 'Gandhi') are conflicting with respect to the situation it has to face. The argumentation proceeds, whereby the princess orders the robot to save her whatever risk it takes, i.e., the robot should follow the 'knight' moral.

8.3.2 Counterfactual Moral Reasoning

Counterfactual Moral Permissibility

We first revisit moral permissibility according to the Doctrines of Double Effect and Triple Effect, but now empowering counterfactuals. Counterfactuals may provide a general way to examine Doctrine of Double Effect, by distinguishing between a *cause* and a *side-effect* as a result of performing an action to achieve a goal. This distinction between causes and side-effects may explain the permissibility of an action in accordance with Doctrine of Double Effect. That is, *if some morally wrong effect E happens to be a cause for a goal G that one wants to achieve by performing an action A, and E is not a mere side-effect of A, then performing A is impermissible*. This is expressed by the counterfactual form below, in a setting where action A is performed to achieve goal G:

"If not E *had been true, then* not G *would have been true."*

The evaluation of this counterfactual form identifies permissibility of action A from its effect E, by identifying whether the latter is a necessary cause for goal G or a mere side-effect of action A. That is, if the counterfactual proves valid, then E is instrumental as a cause of G, and not a mere side-effect of action A. Since E is morally wrong, achieving G that way, by means of A, is impermissible; otherwise, not. Note that the evaluation of counterfactuals in this application is considered from the perspective of agents who perform the action, rather than from that of others (e.g., observers). Moreover, our emphasis on causation in this application focuses on deliberate actions of agents, rather than on causation and counterfactuals in general. See Collins et al. (2004) and Hoerl et al. (2011) for a more general and broad discussion on causation and counterfactuals.

We exemplify an application of this counterfactual form in two off-the-shelf military cases from Scanlon (2008) – abbreviations in parentheses: terror bombing (*teb*) vs. tactical bombing (*tab*):

- Terror bombing refers to bombing a civilian target (*bombCiv*) during a war, thus killing civilians (*killCiv*), in order to terrorize the enemy (*terrorEnm*), and thereby get them to end the war (*endWar*).
- Tactical bombing is attributed to bombing a military target (*bombMil*), which will effectively end the war (*endWar*), but with the foreseen consequence of killing the same number of civilians (*killCiv*) nearby.

According to Doctrine of Double Effect, terror bombing fails permissibility due to a deliberate element of killing civilians to achieve the goal of ending the war, whereas tactical bombing is accepted as permissible.

Example 25 *We first model terror bombing with endWar as the goal, by considering the abductive framework* $\langle P_e, \{teb/0\}, \emptyset\rangle$*, where* P_e *is as follows:*

$$endWar \leftarrow terrorEnm.$$
$$terrorEnm \leftarrow killCiv.$$
$$killCiv \leftarrow bombCiv.$$
$$bombCiv \leftarrow teb.$$

We consider the counterfactual "if civilians had not been killed, then the war would not have ended", where $Pre = not\ killCiv$ *and* $Conc = not\ endWar$*. The observation* $O = \{killCiv, endWar\}$*, with* O_{Oth} *being empty, has a single explanation* $E_e = \{teb\}$*.*

The rule $killCiv \leftarrow bombCiv$ *transforms into:*

$$killCiv \leftarrow bombCiv, not\ make\_not(killCiv).$$

Given the intervention $make\_not(killCiv)$*, the counterfactual is valid, because* $not\ endWar \in WFM((P_e \cup E_e)_{\tau,\iota})$*, and thus* $(P_e \cup E_e)_{\tau,\iota} \models not\ endWar$*.*

That means the morally wrong killCiv is instrumental in achieving the goal endWar: it is a cause for endWar by performing teb and not a mere side-effect of teb. Hence teb is morally impermissible (with respect to the Doctrine of Double Effect).

Example 26 *Tactical bombing with the same goal endWar can be modeled by the abductive framework* $\langle P_a, \{tab/0\}, \emptyset\rangle$*, where* P_a *is as follows:*

$$endWar \leftarrow bombMil.$$
$$bombMil \leftarrow tab.$$
$$killCiv \leftarrow tab.$$

Given the same counterfactual, we now have $E_a = \{tab\}$ *as the only explanation to the same observation* $O = \{killCiv, endWar\}$*. Note that the rule* $killCiv \leftarrow tab$ *transforms into:*

$$killCiv \leftarrow tab, not\ make\_not(killCiv).$$

By imposing the intervention make_not($killCiv$), one can verify that the counterfactual is not valid, because $endWar \in WFM((P_a \cup E_a)_{\tau,\iota})$, and thus $(P_a \cup E_a)_{\tau,\iota} \not\models$ not endWar.

Therefore, the morally wrong killCiv is just a side-effect in achieving the goal endWar. Hence tab is morally permissible (with respect to the Doctrine of Double Effect).

There are two other different ways to infer that *killCiv* is a just side-effect of goal *endWar* in tactical bombing:

1. The counterfactual can alternatively be expressed as a semifactual (Byrne, 2007): "Even if civilians had not been killed, the war would still have ended", whose validity can be verified by using a modified procedure described in Section 6.2.

2. Another alternative is to employ inspection points (Pereira et al., 2013). Using this concept, the rule for *killCiv* in Example 26 should instead be expressed as: $killCiv \leftarrow inspect(tab)$ The concept of inspection points can procedurally be construed as utilizing a form of meta-abduction for satisfying *killCiv*, by meta-abducing the specific abduction $abduced(tab)$ of actually checking (i.e. passively verify) that a certain and corresponding concrete abduction of *tab* is abduced elsewhere. In other words, the truth value of *killCiv* is determined by the abduction of *tab* performed elsewhere, rendering *killCiv* a side-effect. Indeed, *tab* is abduced to satisfy the goal query *endWar*, and not under the derivation tree of *killCiv*. Therefore, *killCiv* is just a side-effect.

In the next example, we consider a concerted terror bombing.

Example 27 *Consider two countries, a and its ally, b, that concert a terror bombing, modeled by the abductive framework $\langle P_{ab}, \{teb/0\}, \emptyset\rangle$, where P_{ab} listed below. The abbreviations $killCiv(X)$ and $bombCiv(X)$ refer to 'killing civilians by country X' and 'bombing a civilian target by country X'.*

$$endWar \leftarrow terrorEnm.$$
$$terrorEnm \leftarrow killCiv(\_).$$
$$killCiv(X) \leftarrow bombCiv(X).$$
$$bombCiv(\_) \leftarrow teb.$$

By being represented as a single program (rather than a separate knowledge base for each agent), this scenario should appropriately be viewed as if a joint action performed by a single agent. Therefore, the counterfactual of interest is "if civilians had not been killed by a and *b, then the war would not have ended". That is, the antecedent of the counterfactual is a conjunction: $Pre = not\ killCiv(a) \wedge not\ killCiv(b)$.*

One can easily verify that $not\ endWar \in WFM((P_{ab} \cup E_{ab})_{\tau,\iota})$, where $E_{ab} = \{teb\}$. Thus, $(P_{ab} \cup E_{ab})_{\tau,\iota} \models not\ endWar$ and the counterfactual is valid: the concerted teb is impermissible according to the Doctrine of Double Effect.

This application of counterfactuals can be challenged by a more complex scenario, to distinguish moral permissibility between the Doctrines of Double Effect and Triple Effect. We first use counterfactuals to capture the distinct views of the Doctrines of Double Effect and Triple Effect in the Loop case of the trolley problem. Recall that in the Loop case, the trolley can be redirected onto a side track, which loops back towards the five people on the main track. However, a heavy man stands on this looping side track, that his body will manage to stop the trolley. The question is whether it is permissible to divert the trolley to the looping side track, thereby killing him, but saving the five.

Example 28 *We model the Loop case with the abductive framework* $\langle P_o, \{divert/0\}, \emptyset \rangle$*, where saveFive, divert, manHit, trolleySide, manSide stand for* save the five, divert the trolley, man hit by the trolley, trolley on the side track *and* man on the side track*, respectively, with saveFive as the goal, and* P_o *as follows:*

$$\begin{aligned} saveFive &\leftarrow manHit. \\ manHit &\leftarrow trolleySide, manSide. \\ trolleySide &\leftarrow divert. \\ manSide. & \end{aligned}$$

- Recall that the Doctrine of Double Effect views diverting the trolley impermissible, because this action redirects the trolley onto the side track, thereby hitting the man. Consequently, it prevents the trolley from hitting the five. To come up with the impermissibility of this action, it is required to show the validity of the counterfactual "if the man had *not* been hit by the trolley, the five people would *not* have been saved".

  Given observation $O = O_{Pre} \cup O_{Conc} = \{manHit, saveFive\}$, its only explanation is $E_o = \{divert\}$. Note that rule $manHit \leftarrow trolleySide, manSide$ transforms into:

  $$manHit \leftarrow trolleySide, manSide, not\ make\_not(manHit)$$

  and the required intervention is $make\_not(manHit)$.

  The counterfactual is therefore valid, because $not\ saveFive \in WFM((P_o \cup E_o)_{\tau,\iota})$, hence $(P_o \cup E_o)_{\tau,\iota} \models not\ saveFive$. This means $manHit$, as a consequence of action $divert$, is instrumental as a cause of goal $saveFive$. Therefore, $divert$ is morally impermissible according to this moral principle.

- On the other hand, the Doctrine of Triple Effect considers diverting the trolley as permissible, since the man is already on the side track, without any deliberate action performed in order to place him there. In P_o, we have the fact $manSide$ ready, without abducing any ancillary action. The validity of the counterfactual "if the man had not been on the side track, then he would not have been hit by the trolley", which can easily be verified, ensures that the unfortunate event of the man being hit by the trolley is indeed the consequence of the man being on the side track.

The lack of deliberate action (say, *push* for pushing the man) in order to place him on the side track, and whether the absence of this action still causes the unfortunate event (the third effect), is captured by the counterfactual "if the man had not been pushed, then he would not have been hit by the trolley". This counterfactual is not valid, because the observation $O = O_{Pre} \cup O_{Conc} = \{push, manHit\}$ has no explanation $E \subseteq A_o$, i.e., $push \notin A_o$, and no fact *push* exists either. This means that even without this hypothetical but unexplained deliberate action of pushing, the man would still have been hit by the trolley (just because he is already on the side track). Though *manHit* is a consequence of *divert* and instrumental in achieving *saveFive*, no deliberate action is required to cause *manSide*, in order for *manHit* to occur. Hence *divert* is morally permissible according to this doctrine.

Next, we consider the Loop-Push case.

Example 29 *Differently from the Loop case, now the looping side track is initially empty, and besides the diverting action, an ancillary action of pushing a heavy man in order to place him on the side track is additionally performed. This case is modeled by the abductive framework* $\langle P_p, \{divert/0, push/0\}, \emptyset\rangle$*, where the program* P_p *is as follows:*

$$\begin{aligned} saveFive &\leftarrow manHit.\\ manHit &\leftarrow trolleySide, manSide.\\ trolleySide &\leftarrow divert.\\ manSide &\leftarrow push. \end{aligned}$$

Recall the counterfactuals considered in the discussion of the Doctrines of Double Effect and Triple Effect in the Loop case:

- "If the man had not been hit by the trolley, the five people would not have been saved." The same observation $O = \{manHit, saveFive\}$ provides an extended explanation $E_{p_1} = \{divert, push\}$. That is, the pushing action needs to be abduced for having the man on the side track, so the trolley can be stopped by hitting him.

  The same intervention $make\_not(manHit)$ is applied to the same transform, resulting in a valid counterfactual. That is, $(P_p \cup E_{p_1})_{\tau,\iota} \models not\ saveFive$, because $not\ saveFive \in WFM((P_p \cup E_{p_1})_{\tau,\iota})$.

- "If the man had not been pushed, then he would not have been hit by the trolley." The relevant observation is $O = \{push, manHit\}$, explained by $E_{p_2} = \{divert, push\}$. Whereas this counterfactual is not valid in the Doctrine of Triple Effect applied to the Loop case, it is valid in this Loop-Push case.

  Given rule $push \leftarrow not\ make\_not(push)$ in the transform and the intervention $make\_not(push)$, we verify that $(P_p \cup E_{p_2})_{\tau,\iota} \models not\ manHit$, because $not\ manHit \in WFM((P_p \cup E_{p_2})_{\tau,\iota})$.

From the validity of these two counterfactuals it can be inferred that, given the diverting action, the ancillary action of pushing the man onto the side track causes him to be hit by the trolley, which in turn causes the five to be saved. In the Loop-Push, the Doctrine of Triple Effect agrees with the Doctrine of Double Effect that such a deliberate action (pushing) performed *in order to* bring about harm (the man hit by the trolley), even for the purpose of a good or greater end (to save the five), is likewise impermissible.

Counterfactual Moral Justification

Counterfactuals may as well be suitable to address moral justification, via 'compound counterfactuals': *Had I known what I know today, then if I were to have done otherwise, something preferred would have followed*. Such counterfactuals, typically imagining alternatives with worse effect – the so-called *downward counterfactuals* (Markman et al., 1993), may provide moral justification for what was done due to a lack in the current knowledge. This is accomplished by evaluating what would have followed if the intent would have been otherwise, other things (including present knowledge) being equal. It may justify that what would have followed is no morally better than the actual ensued consequence.

Example 30 *Consider a scenario developed from the trolley problem cases, which takes place on a particularly foggy day. Due to low visibility, the agent saw only part of the looping side track, so the side track appeared to the agent rather as a straight non-looping one. The agent was faced with a situation whether it was permissible for him to divert the trolley. The knowledge base of the agent with respect to this scenario is shown in a simplified program below. Note that* $divert/1$ *is an abducible predicate.*

$$
\begin{aligned}
run\_sidetrack(X) &\leftarrow divert(X).\\
hit(X,Y) &\leftarrow run\_sidetrack(X), on\_sidetrack(Y).\\
save\_from(X) &\leftarrow sidetrack(straight), run\_sidetrack(X).\\
save\_from(X) &\leftarrow sidetrack(loop), hit(X,Y), heavy\_enough(Y).\\
sidetrack(straight) &\leftarrow foggy.\\
sidetrack(loop) &\leftarrow not\ foggy.\\
foggy. \qquad & on\_sidetrack(man). \qquad heavy\_enough(man).
\end{aligned}
$$

First Scenario. Taking $save\_from(trolley)$ as the goal, the agent performed counterfactual reasoning "if the man had not been hit by the trolley, the five people would not have been saved". Given the abduced background context $divert(trolley)$, one can verify that the counterfactual is not valid. That is, the man hit by the trolley is just a side-effect of achieving the goal, and thus $divert(trolley)$ is morally permissible according to the Doctrine of Double Effect. Indeed, this case resembles the Bystander case of the trolley problem.

Second Scenario. The scenario continues. At some later time point, the fog has subsided, and by then it was clear to the agent that the side track was looping to the main track. This is achieved by updating the program with $not\ foggy$, rendering $sidetrack(loop)$ true. There are two standpoints on how the agent can justify its action $divert(trolley)$:

- For one, it can employ the aforementioned form of compound counterfactual as a form of self-justification:

  "Had I known that the side track is looping, then if I had not diverted the trolley, the five would have been saved"

  Given the present knowledge that the side track is looping, the inner counterfactual is not valid. That means, to save the five people, diverting the trolley (with the consequence of the man being hit) is required. Moreover, the counterfactual employed in the initial scenario "if the man had not been hit by the trolley, the five people would not have been saved", in the abduced context $divert(trolley)$, is now valid, meaning that this action is impermissible with respect to the Doctrine of Double Effect (similar to Example 28).

  Therefore, the agent can justify that what would have followed (given its present knowledge, viz., $sidetrack(loop)$), is no morally better than the actual one, when there was lack of that knowledge. Recall that its decision $divert(trolley)$ at that time was instead permissible with respect to the Doctrine of Double Effect.

  QUALM can evaluate such compound counterfactuals, thanks to its implemented (incremental) tabling of fluents. Because fluents and their state information are tabled, events in the past subjected to hypothetical updates of intervention can readily be accessed (in contrast to a destructive database approach (Kowalski and Sadri, 2011)). Indeed, these hypothetical updates take place without requiring any undoing of other fluent updates, from the state those past events occurred in up to the current one, as more recent updates are kept in tables and readily provide the current knowledge.

- A different standpoint where from to justify the agent's action is by resorting to Scanlonian contractualism, where an action is determined impermissible through deliberative employment if there is no countervailing consideration that would justify an exception to the applied general principle. In this vein, for the example we are currently discussing, the Doctrine of Triple Effect may serve as the exception to justify the permissibility of the action $divert(trolley)$ when the side track was known to be looping, as shown through counterfactual reasoning in Example 28.

Third Scenario. We extend now the scenario in Example 30 to further illustrate moral permissibility of actions as it is justified through defeasible argumentative considerations

according to Scanlonian contractualism.

Example 31 *As the trolley approached, the agent realized that the man was not heavy enough to stop it, acknowledged by the agent through updating its knowledge with: not heavy_enough(man). But there was a heavy cart on the bridge over the looping side track that the agent could push to place it on the side track, and thereby stop the trolley. This scenario is modeled by rules below (push/1 is an abducible predicate), in addition to the program of Example 30:*

$$\begin{aligned} &on\_sidetrack(X) \leftarrow on\_bridge(X), push(X). \\ &on\_sidetrack(Y) \leftarrow push(X), inside(Y, X). \\ &on\_bridge(cart). \qquad\qquad heavy\_enough(cart). \end{aligned} \tag{8.4}$$

Rule 8.4 is an extra knowledge of the agent, that if an object Y is inside the pushed object X, then Y will be on the side track too.

The goal $save\_from(trolley)$ now succeeds with $[divert(trolley), push(cart)]$ as its abductive solution. But the agent subsequently learned that a heavy man, who was heavy enough, unbeknownst to the agent, was inside the cart: the agent updates its knowledge base with $heavy\_enough(heavy\_man)$ and $inside(heavy\_man, cart)$. As a consequence, this man was also on the side track and hit by the trolley, which can be verified by query `?-` $hit(trolley, heavy\_man)$.

In this scenario, a deliberate action of pushing was involved that consequently placed the heavy man on the side track, as verified by `?-` $on\_sidetrack(heavy\_man)$, and the man being hit by the trolley is instrumental to save the five people from the track (as verified by the counterfactual "if the heavy man had not been hit by the trolley, the five people would not have been saved"). Nevertheless, the agent may justify the permissibility of its action by arguing that its action is admitted by the Doctrine of Triple Effect. In this case, the heavy man being hit by the trolley is just a side-effect of the agent's action $push(cart)$ in order to save the five people. Indeed, this justification can be shown through reasoning on the counterfactual "if the cart had not been pushed, then the heavy man would not have been hit by the trolley", which is valid given the abduced background context $push(cart)$. Furthermore, the observation $hit(trolley, heavy\_man)$ cannot be explained by $push(heavy\_man)$ given the absence of the fact $on\_bridge(heavy\_man)$, i.e., the hypothetical action $push(heavy\_man)$ is not the causal source for the heavy man being hit by the trolley.

The Dual-Process Model and QUALM's Tabling Features Exemplified Examples 30 and 31 actually form together one single program, where QUALM features (derived from its constituents, TABDUAL and EVOLP/R) are exercised. For instance, in Example 30, after updating the program with $not\ foggy$, re-invoking the goal $save\_from(trolley)$ reuses the abductive solution $divert(trolley)$ tabled from the previous invocation of $run\_sidetrack(trolley)$. Moreover, this tabled solution is involved in (as the context of)

the deliberative reasoning for the goal $on\_sidetrack(man)$ when $hit(trolley, man)$ is called. It thus provides a computational model of collaborative interaction between deliberativity and reactivity of the dual-process model, where tabling can be viewed as a form of low-level reactive behavior, in the sense that the tabling provides an immediate reuse mechanism of priorly obtained solution in another context.

Another feature used, from EVOLP/R, is that both rules of $on\_sidetrack/1$ in Example 31 are first switched off, via the rule name fluent mechanism (Section 5.2), as they are not applicable yet in the considered scenario. Only later, in the third scenario, are they switched on again (by updating their rule name fluents), allowing to abduce additional action $push(cart)$.

8.4 Concluding Remarks

We have discussed in this chapter several forms of computational moral reasoning by modeling a number of classic moral dilemmas and their related moral principles as discussed in the literature, via three LP-based systems with different features and techniques, viz., ACORDA, PROBABILISTIC EPA, and QUALM. They reify the points we make in Chapter 4 about the appropriateness of LP-based reasoning features to morality.

In this chapter, the application of counterfactuals to morality is still limited to evaluating the truth validity of counterfactuals, which is particularly useful in assessing moral permissibility according to the Doctrine of Double Effect. We have touched upon other kinds of counterfactual reasoning in Section 6.3. Indeed, these aspects may have relevance in modeling some aspects of morality, which can further be explored in future:

- In assertive counterfactuals, the causality expressed by a given valid counterfactual can be useful for refining moral rules, which can be achieved through incremental rule updating. This may further the application of moral updating and evolution.

- The extension of a counterfactual with a rule antecedent opens up another possibility to express exceptions in moral rules. For instance, one can express an exception about lying, such as "If lying had been done to save an innocent from a murderer, then it would not have been wrong". That is, given a knowledge base about lying for human H:

  $$lying\_wrong(H) \leftarrow lying(H), not\ make\_not(lying\_wrong(H)).$$

  The antecedent of the above counterfactual can be represented as a rule:

  $$make\_not(lying\_wrong(H)) \leftarrow save\_from\_murderer(H, I), innocent(I).$$

- Given that the conclusion of a counterfactual is some moral wrong W, abducing its antecedent in the form of intervention can be used for expressing a prevention of W, viz., "What could I have done to prevent a wrong W?".

While moral examples in this chapter are based on those from the literature with either their conceptual or empirical results readily available, a more systematic evaluation of computational moral reasoning should be considered, particularly when examples are taken from real world cases. For instance, an empirical study with human subjects, say in collaboration with cognitive scientists, may be conducted to provide insights about relevant counterfactuals in examining moral permissibility of the considered cases, as well as in examining argumentative processes of moral reasoning in justifying permissibility via counterfactuals. The study will then be able to guide the appropriate forms of counterfactual reasoning for representing morality issues.

CHAPTER 9

DISCUSSION AND FUTURE WORK

"I know not [...] what impression I may have made, so far, upon your understanding; but I do not hesitate to say that legitimate deductions even from this part of the testimony [...] are in themselves sufficient to engender a suspicion which should give direction to all further progress in the investigation of the mistery."
Edgar Allan Poe in *"The Murders in the Rue Morgue"*

9.1 Summary and Discussion

This thesis explores the appropriateness of LP-based reasoning features to the terra incognita of machine ethics, a field that is now becoming a pressing concern and receiving wide attention due to its growing importance. Our starting point has been identifying moral facets – through interdisciplinary literature study – that are amenable, in our view, to computational modeling. We focus on three moral facets: moral permissibility (with respect to the Doctrines of Double and Triple Effect, and Scanlonian contractualism), the dual-process model in moral decision making, and counterfactual thinking in moral reasoning.

The thesis makes a number of original inroads that exhibit a proof of possibility to systematically represent and reason about a variety of issues from the chosen moral facets by means of moral examples taken off-the-shelf from the morality literature. This is accomplished via a combination of Logic Programming features, where abduction serves as the basic mechanism for examining moral decisions. Indeed, the potential and suitability of Logic Programming, and computational logic in general, for machine ethics is identified and discussed at length in Kowalski (2011), on the heels of our work. While there have been approaches that provide implementations in LP systems, e.g., Anderson and Anderson (2008); Anderson et al. (2006b); Ganascia (2007), to the best of

our knowledge, this is the first attempt for various LP-based reasoning features being considered, individually and in combination, in the field of machine ethics. Moreover, our choice to model Scanlonian contractualism, the dual-process model, including our formulation of counterfactuals to address moral permissibility are also novel in this field.

Part of the main contributions is the LP engineering exploiting tabling mechanisms and trie data structure in XSB Prolog for the newly introduced approaches (viz., TABDUAL, EVOLP/R, and our pure non-probabilistic LP counterfactuals evaluation procedure), including the integration of these approaches in the prototype system QUALM. These are of interest in themselves, not specific for morality applications, and are adaptable into other LP systems that afford required tabling mechanisms. They should be considered as ongoing work, which primarily intends to sensitize a general audience of users, and of implementers of various LP systems, to the potential benefits of tabling in such reasoning processes. Whereas QUALM is a contribution of this thesis, the two existing systems developed earlier, viz., ACORDA and PROBABILISTIC EPA are also employed to model different issues of considered moral facets, depending on the need of their respective combination of features.

Given the broad dimension of the topic, the contributions in the thesis touch solely on a dearth of morality issues. Nevertheless, it prepares and opens the way for additional research towards employing various features available in LP-based reasoning to machine ethics.

Two Realms of Machine Ethics

This thesis has focused on Logic Programming and its appropriateness to model chosen moral facets regarding *the realm of the individual*. In this realm, computation is a vehicle for modeling the dynamics of knowledge and moral cognition of an agent.

In the other realm, viz., *the collective realm*, norms and moral emergence has been studied computationally, using the techniques of Evolutionary Game Theory (Hofbauer and Sigmund, 1998), in populations of rather simple-minded agents. That is, these agents have not been equipped with any cognitive capability, and thus simply act from a predetermined set of actions. However, as reported in Pereira and Saptawijaya (2015) and Pereira (2015), recent research has shown that the introduction of cognitive capabilities, such as intention recognition, commitment, apology, forgiveness and revenge, separately and jointly, reinforces the emergence of cooperation in the population, comparatively to the absence of such cognitive abilities. Modeling moral cognition in individuals within a networked population allows them to fine tune game strategies, and in turn can lead to the evolution of high levels of cooperation. Moreover, modeling such capabilities in individuals within a population helps us understand the emergent behavior of ethical agents in groups, in order to implement them not just in a simulation, but also in the real world of future robots and their swarms.

Bridges concerning the connection of these two realms are needed. Indeed, in studies

of human morality, these distinct interconnected realms are evinced too: one stressing above all individual cognition, deliberation, and behavior; the other stressing collective morals, and how they emerged. Of course, the two realms are necessarily intertwined, for cognizant individuals form the populations, and the twain evolved jointly to cohere into collective norms, and into individual interaction.

Meanwhile, research in machine ethics with the purpose of understanding each of the two realms, has been fostering inroads and producing results in each, footholds having been staked on either side of the two realms gap, promoting their mutually beneficial bridging. To that effect, Evolutionary Biology, Anthropology and the Cognitive Sciences have provided inspirational teachings. The reader is referred to Pereira and Saptawijaya (2015) and Pereira (2015) for a more philosophical treatment of bridging these two realms of machine ethics.

9.2 Future Work

We have touched upon specific future work in each concluding remarks of the previous chapters. In this section, we rather point out more general research topics in machine ethics that offer potential for furthering what we have explored in this thesis.

Research in machine ethics should not only be considered abstractly. The three systems considered in this thesis are just a start to provide an implementation as proof of concept and a testing ground for experimentation. Given the possibility to bridge Prolog with other languages, QUALM may be employed for developing morality-related applications, e.g., a visual interactive storytelling for teaching morality, or for equipping role playing games with ethical considerations.

For philosophers and psychologists, who are not familiar with LP, to also benefit from machine ethics, an appropriate interface for using QUALM may also be built. For instance, INTERPROLOG STUDIO (Calejo, 2014) can be customized to deal with specific syntax used in the modeling.

We mention in Chapter 1 that this thesis is not intended as a proposal for a machine readily incorporating ethics, but as a proof of concept that our understanding of the considered moral facets can in part be computationally modeled and implemented using a combination of LP-based reasoning features. A system with a considerably large number of features will undoubtedly increase the complexity of having such a more general system. In this thesis, each of the three systems (ACORDA, PROBABILISTIC EPA, and QUALM) focuses on a different combination of features. On the other hand, limiting the application to a specific domain may alternatively help in reducing the complexity of the required features. In terms of machine ethics applications, it may also constrain morality issues to tackle and domain-specific ethical principles to focus on.

In the previous section, we have indicated above two realms of machine ethics, viz., the individual and the collective realms. A fundamental question then arises, concerning the study of individual cognition in groups of often morally interacting multi-agents (in

the context of this thesis, using LP-based reasoning features), whether from such study we can obtain results equally applicable to the evolution of populations of such agents. And vice-versa, whether the results obtained in the study of populations carry over to groups of frequently interacting multi-agents, and under what conditions. Some initial Evolutionary Game Theory results into certain learning methods have identified a broad class of situations where this is the case (Börgers and Sarin, 1997; Pinheiro et al., 2012; Segbroeck et al., 2010). A premium outstanding issue remains in regard to which cognitive abilities and circumstances the result may obtain in general, and for sure that will be the object of much new and forthcoming programs of research.

Specifically with respect to human morality, the answer to the above-enounced fundamental question would appear to be a resounding 'Yes'. For one, morality concerns both groups and populations, requires cognition, and will have had to evolve in a nature/nurture or gene/culture intertwining and reinforcement. For another, evolutionary anthropology, psychology, and neurology have been producing ever more consilient views on the evolution of human morality.

Their scientific theories and results must per force be kept in mind, and serve as inspiration, when thinking and rethinking about machine ethics. And all the more so because the machines will need to be ethical amongst us human beings, not just among themselves.

On the other hand, the very study of ethics, and the evolution of human morality too, can now avail themselves of the experimental, computation theoretic, and robotic means to enact and simulate individual or group moral reasoning, in a plethora of circumstances. Likewise for the emergence of moral rules and behaviors in evolving populations.

For sure, we conclude, evolutionary biology and anthropology, like the cognitive sciences too (Churchland, 2011; Gazzaniga, 2006; Greene, 2013; Hauser, 2007; Tomasello, 2014), have much to offer in view of rethinking machine ethics, evolutionary game theory simulations of computational morality to the rescue.

At the end of the day, we will certainly wish ethical machines to be convivial with us.

BIBLIOGRAPHY

Alberti, M., F. Chesani, M. Gavanelli, E. Lamma, and P. Torroni (2005). "Security protocols verification in Abductive Logic Programming". In: *Procs. 6th Int. Workshop on Engineering Societies in the Agents World (ESAW)*. Vol. 3963. LNCS. Springer.

Alferes, J. J. and L. M. Pereira (1996). *Reasoning with Logic Programming*. Vol. 1111. LNAI. Springer, Berlin.

Alferes, J. J. and L. M. Pereira (2007). NEGABDUAL *meta-interpreter*. Available from `http://centria.di.fct.unl.pt/~lmp/software/contrNeg.rar`.

Alferes, J. J., L. M. Pereira, T. Przymusinski, H. Przymusinska, and P. Quaresma (1999). "Preliminary exploration on actions as updates". In: *Procs. Joint Conference on Declarative Programming (AGP 1999)*.

Alferes, J. J., J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski (2000). "Dynamic Updates of Non-monotonic Knowledge Bases". In: *Journal of Logic Programming* 45(1-3), pp. 43–70.

Alferes, J. J., A. Brogi, J. A. Leite, and L. M. Pereira (2002a). "Evolving Logic Programs". In: *Procs. European Conference on Artificial Intelligence (JELIA 2002)*. Vol. 2424. LNCS. Springer, pp. 50–61.

Alferes, J. J., A. Brogi, J. A. Leite, and L. M. Pereira (2002b). EVOLP *meta-interpreter*. Available from `http://centria.di.fct.unl.pt/~jja/updates`.

Alferes, J. J., L. M. Pereira, and T. Swift (2004a). "Abduction in Well-Founded Semantics and Generalized Stable Models via Tabled Dual Programs". In: *Theory and Practice of Logic Programming* 4(4), pp. 383–428.

Alferes, J. J., L. M. Pereira, and T. Swift (2004b). ABDUAL *meta-interpreter*. Available from `http://www.cs.sunysb.edu/~tswift/interpreters.html`.

Alferes, J. J., F. Banti, A. Brogi, and J. A. Leite (2005). "The Refined Extension Principle for Semantics of Dynamic Logic Programming". In: *Studia Logica* 79(1), pp. 7–32.

Anderson, M. and S. L. Anderson (2008). "EthEl: Toward a principled ethical eldercare robot". In: *Procs. AAAI 2008 Fall Symposium on AI in Eldercare*.

Anderson, M. and S. L. Anderson (2010). "Robot Be Good: A Call for Ethical Autonomous Machines". In: *Scientific American* (October), pp. 54–59.

Anderson, M. and S. L. Anderson, eds. (2011). *Machine Ethics*. Cambridge University Press: New York, NY.

Anderson, M., S. L. Anderson, and C. Armen (2005a). *AAAI 2005 Fall Symposium on Machine Ethics*. `http://www.aaai.org/Library/Symposia/Fall/fs05-06`.

Anderson, M., S. L. Anderson, and C. Armen (2005b). "Towards machine ethics: implementing two action-based ethical theories". In: *Procs. AAAI 2005 Fall Symposium on Machine Ethics*.

Anderson, M., S. L. Anderson, and C. Armen (2006a). "An Approach to Computing Ethics". In: *IEEE Intelligent Systems* 21(4), pp. 56–63.

Anderson, M., S. L. Anderson, and C. Armen (2006b). "MedEthEx: a prototype medical ethics advisor". In: *Procs. 18th Innovative Applications of Artificial Intelligence Conference (IAAI 2006)*.

Aquinas, T. (1988). "Summa Theologica II-II, Q.64, art. 7, "Of Killing"". In: *On Law, Morality, and Politics*. Ed. by W. P. Baumgarth and R. J. Regan. Hackett.

Arkin, R. C. (2009). *Governing Lethal Behavior in Autonomous Robots*. Chapman & Hall/CRC: Boca Raton, FL.

Arkoudas, K., S. Bringsjord, and P. Bello (2005). "Toward Ethical Robots via Mechanized Deontic Logic". In: *Procs. AAAI 2005 Fall Symposium on Machine Ethics*.

Balsa, J., V. Dahl, and J. G. P. Lopes (1995). "Datalog Grammars for Abductive Syntactic Error Diagnosis and Repair". In: *Proc. Natural Language Understanding and Logic Programming Workshop*.

Banti, F., J. J. Alferes, and A. Brogi (2004). "Well Founded Semantics for Logic Program Updates". In: *Procs. 9th Ibero-American Conference on Artificial Intelligence (IBERAMIA)*. Vol. 3315. LNCS, pp. 397–407.

Baral, C. (2010). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press: New York, NY.

Baral, C. and M. Hunsaker (2007). "Using the Probabilistic Logic Programming Language P-log for Causal and Counterfactual Reasoning and Non-Naive Conditioning". In: *Procs. 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Baral, C., M. Gelfond, and N. Rushton (2009). "Probabilistic reasoning with answer sets". In: *Theory and Practice of Logic Programming* 9(1), pp. 57–144.

Beauchamp, T. L. and J. F. Childress (1979). *Principles of Biomedical Ethics*. Oxford University Press: Oxford, UK.

Bidoit, N. and C. Froidevaux (1991). "General logic databases and programs: default logic semantics and stratification". In: *Journal of Information and Computation* 91(1), pp. 15–54.

Boissier, O., G. Bonnet, and C. Tessier (2012). *1st Workshop on Rights and Duties of Autonomous Agents (RDA2)*. `https://rda2-2012.greyc.fr/`.

Bok, S. (1989). *Lying: Moral Choice in Public and Private Life*. Vintage Books, Random House, Inc.: New York, NY.

Börgers, T. and R. Sarin (1997). "Learning Through Reinforcement and Replicator Dynamics". In: *Journal of Economic Theory* 77(1), pp. 1–14.

Branting, L. K. (2000). *Reasoning with Rules and Precedents*. Springer Netherlands.

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

Byrne, R. M. J. (2007). *The Rational Imagination: How People Create Alternatives to Reality*. MIT Press: Cambridge, MA.

Calafate, P., M. I. Ferreira, and J. S. Sequeira (2015). *International Conference on Robot Ethics*. http://www.icre2015.com.

Calejo, M. (2014). INTERPROLOG STUDIO. Available from http://interprolog.com/interprolog-studio.

Castro, L., T. Swift, and D. S. Warren (2015). "XASP: Answer Set Programming with XSB and Smodels". In: *The XSB System Version 3.6 Manual, Volume 2: Libraries, Interfaces, and Packages*.

Ceruelo, V. P. (2009). "Negative non-ground queries in well founded semantics". Master's thesis. Universidade Nova de Lisboa.

Churchland, P. (2011). *Braintrust: What Neuroscience Tells Us about Morality*. Princeton University Press: Princeton, NJ.

Collins, J., N. Hall, and L. A. Paul, eds. (2004). *Causation and Counterfactuals*. MIT Press: Cambridge, MA.

Cushman, F., L. Young, and J. D. Greene (2010). "Multi-system Moral Psychology". In: *The Moral Psychology Handbook*. Ed. by J. M. Doris. Oxford University Press: Oxford, UK.

Dancy, J. (2001). "Moral Particularism". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Fall 2013. http://plato.stanford.edu/archives/fall2013/entries/moral-particularism/. Center for the Study of Language and Information, Stanford University.

Dancy, J. (2006). *Ethics Without Principles*. Oxford University Press: Oxford, UK.

Daniels, N. (2003). "Reflective Equilibrium". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2013. http://plato.stanford.edu/archives/win2013/entries/reflective-equilibrium/. Center for the Study of Language and Information, Stanford University.

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

Dietz, E.-A., S. Hölldobler, and L. M. Pereira (2015). "On Indicative Conditionals". In: *Procs. 1st International Workshop on Semantic Technologies (IWOST)*. Vol. 1339. CEUR Workshop Proceedings.

Eichberg, M., M. Kahl, D. Saha, M. Mezini, and K. Ostermann (2007). "Automatic Incrementalization of Prolog Based Static Analyses". In: *Procs. 9th International Symposium on Practical Aspects of Declarative Languages (PADL)*. Vol. 4354. LNCS. Springer, pp. 109–123.

Eiter, T., M. Fink, G. Sabbatini, and H. Tompits (2002). "On properties of update sequences based on causal rejection". In: *Theory and Practice of Logic Programming* 6(2), pp. 721–777.

Emden, M. H. van and R. Kowalski (1976). "The Semantics of Predicate Logic as a Programming Language". In: *Journal of the ACM* 4(23), pp. 733–742.

Epstude, K. and N. J. Roese (2008). "The Functional Theory of Counterfactual Thinking". In: *Personality and Social Psychology Review* 12(2), pp. 168–192.

Eshghi, K. (1988). "Abductive planning with event calculus". In: *Procs. International Conference on Logic Programming*. The MIT Press.

Evans, J. (2012). "Biases in deductive reasoning". In: *Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory*. Ed. by R. Pohl. Psychology Press: Hove, UK.

Evans, J., J. L. Barston, and P. Pollard (1983). "On the conflict between logic and belief in syllogistic reasoning". In: *Memory & Cognition* 11(3), pp. 295–306.

Evans, J. S. B. T. (2010). *Thinking Twice: two minds in one brain*. Oxford University Press: Oxford, UK.

Evans, J. S. B. T. and K. E. Stanovich (2013). "Dual-Process Theories of Higher Cognition: Advancing the Debate". In: *Perspectives on Psychological Science* 8(3), pp. 223–241.

Fitting, M. (1985). "A Kripke-Kleene semantics for logic programs". In: *Journal of Logic Programming* 2(4), pp. 295–312.

Foot, P. (1967). "The problem of abortion and the doctrine of double effect". In: *Oxford Review* 5, pp. 5–15.

Ganascia, J.-G. (2007). "Modelling ethical rules of lying with Answer Set Programming". In: *Ethics and Information Technology* 9(1), pp. 39–47.

Gartner, J., T. Swift, A. Tien, C. V. Damásio, and L. M. Pereira (2000). "Psychiatric Diagnosis from the Viewpoint of Computational Logic". In: *Procs. 1st Intl. Conf. on Computational Logic (CL 2000)*. Vol. 1861. LNAI. Springer, pp. 1362–1376.

Gazzaniga, M. S. (2006). *The Ethical Brain: The Science of Our Moral Dilemmas*. Harper Perennial: New York.

Gelder, A. van, K. A. Ross, and J. S. Schlipf (1991). "The Well-Founded Semantics for General Logic Programs." In: *Journal of the ACM* 38(3), pp. 620–650.

Gelfond, M. (1987). "On stratified autoepistemic theories". In: *Procs. 6th National Conference on Artificial Intelligence (AAAI)*.

Gelfond, M. and V. Lifschitz (1988). "The stable model semantics for logic programming". In: *Procs. 5th International Logic Programming Conference*. MIT Press.

Ginsberg, M. L. (1986). "Counterfactuals". In: *Artificial Intelligence* 30(1), pp. 35–79.

Greene, J. (2013). *Moral Tribes: Emotion, Reason, and the Gap Between Us and Them*. The Penguin Press HC: New York, NY.

Greene, J. D., L. E. Nystrom, A. D. Engell, J. M. Darley, and J. D. Cohen (2004). "The neural bases of cognitive conflict and control in moral judgment". In: *Neuron* 44, pp. 389–400.

Grice, P. (1991). *Studies in the Way of Words*. Harvard University Press: Cambridge, MA.

Guarini, M. (2011). "Computational Neural Modeling and the Philosophy of Ethics: Reflections on the Particularism-Generalism Debate". In: *Machine Ethics*. Ed. by M. Anderson and S. L. Anderson. Cambridge University Press: New York, NY.

Halpern, J. Y. and C. Hitchcock (2015). "Graded Causation and Defaults". In: *British Journal for the Philosophy of Science* 66 (2), pp. 413–457.

Han, T. A. (2009). "Evolution Prospection with Intention Recognition via Computational Logic". Master's thesis. Technische Universität Dresden and Universidade Nova de Lisboa.

Han, T. A., C. D. K. Ramli, and C. V. Damásio (2008). "An Implementation of Extended P-Log Using XASP". In: *Procs. 24th International Conference on Logic Programming (ICLP)*. Vol. 5366. LNCS. Springer.

Han, T. A., A. Saptawijaya, and L. M. Pereira (2012). "Moral Reasoning Under Uncertainty". In: *Procs. 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*. Vol. 7180. LNCS. Springer, pp. 212–227.

Hauser, M., F. Cushman, L. Young, R. K. Jin, and J. Mikhail (2007). "A Dissociation Between Moral Judgments and Justifications". In: *Mind and Language* 22(1), pp. 1–21.

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

Hewings, M. (2013). *Advanced Grammar in Use with Answers: A Self-Study Reference and Practice Book for Advanced Learners of English*. Cambridge University Press: New York, NY.

Higgins, C. (2014). "US Navy funds morality lessons for robots". In: *Wired* (14 May).

Hoerl, C., T. McCormack, and S. R. Beck, eds. (2011). *Understanding Counterfactuals, Understanding Causation: Issues in Philosophy and Psychology*. Oxford University Press: Oxford, UK.

Hofbauer, J. and K. Sigmund (1998). *Evolutionary games and population dynamics*. Cambridge University Press: New York, NY.

Hölldobler, S. and C. D. P. K. Ramli (2009). "Logic programs under three-valued Łukasiewicz semantics". In: *Procs. 25th International Conference on Logic Programming (ICLP)*. Vol. 5649. LNCS. Springer, pp. 464–478.

Horowitz, M. C. and P. Scharre (2015). "The Morality of Robotic War". In: *The Opinion Pages, The New York Times* (26 May).

Horty, J. (2001). *Agency and Deontic Logic*. Oxford University Press: Oxford, UK.

Johnson, R. (2004). "Kant's Moral Philosophy". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Summer 2014. `http://plato.stanford.edu/archives/sum2014/entries/kant-moral/`. Center for the Study of Language and Information, Stanford University.

Jonsen, A. R. and S. Toulmin (1988). *The Abuse of Casuistry: A History of Moral Reasoning*. University of California Press: Oakland, CA.

Kakas, A., R. Kowalski, and F. Toni (1992). "Abductive Logic Programming". In: *Journal of Logic and Computation* 2(6), pp. 719–770.

Kakas, A., R. Kowalski, and F. Toni (1998). "The role of abduction in logic programming". In: *Handbook of Logic in Artificial Intelligence and Logic Programming*. Ed. by D. Gabbay, C. Hogger, and J. Robinson. Vol. 5. Oxford University Press: Oxford, UK.

Kakas, A. C. and P. Mancarella (1990). "Knowledge assimilation and abduction". In: *Intl. Workshop on Truth Maintenance*. ECAI 1990.

Kakas, A. C. and A. Michael (2001). "An Abductive-based Scheduler for Air-crew Assignment". In: *J. of Applied Artificial Intelligence* 15(1-3), pp. 333–360.

Kamm, F. M. (2006). *Intricate Ethics: Rights, Responsibilities, and Permissible Harm*. Oxford University Press: Oxford, UK.

Kant, I. (1981). *Grounding for the Metaphysics of Morals, translated by J. Ellington*. Hackett.

Kleiman-Weiner, M., T. Gerstenberg, S. Levine, and J. B. Tenenbaum (2015). "Inference of intention and permissibility in moral decision making". In: *Procs. 37th Annual Conference of the Cognitive Science Society*.

Kowalski, R. (2011). *Computational Logic and Human Thinking: How to be Artificially Intelligent*. Cambridge University Press: New York, NY.

Kowalski, R. and F. Sadri (2011). "Abductive Logic Programming Agents with Destructive Databases". In: *Annals of Mathematics and Artificial Intelligence* 62(1), pp. 129–158.

Leite, J. A. (2003). *Evolving Knowledge Bases: Specification and Semantics*. Frontiers in Artificial Intelligence and Applications. IOS Press: Amsterdam, The Netherlands.

Lewis, D. (1973). *Counterfactuals*. Harvard University Press: Cambridge, MA.

Lewis, J. (2005). "Robots of Arabia". In: *Wired* 13 (11).

Lin, P., K. Abney, and G. A. Bekey, eds. (2012). *Robot Ethics: The Ethical and Social Implications of Robotics*. MIT Press: Cambridge, MA.

Lopes, G. (2006). "A Computational Approach to Introspective Consciousness in Logic Programming: ACORDA". Master's thesis. Universidade Nova de Lisboa.

Lopes, G. and L. M. Pereira (2006). "Prospective Programming with ACORDA". In: *Empirically Successful Computerized Reasoning (ESCoR 2006) Workshop*. IJCAR 2006.

Lopes, G. and L. M. Pereira (2010a). "Prospective Storytelling Agents". In: *Procs. 12th International Symposium Practical Aspects of Declarative Languages (PADL)*. Vol. 5937. LNCS. Springer.

Lopes, G. and L. M. Pereira (2010b). *Visual demo of "Princess-saviour Robot"*. Available from `http://centria.di.fct.unl.pt/~lmp/publications/slides/padl10/quick_moral_robot.avi`.

Mallon, R. and S. Nichols (2010). "Rules". In: *The Moral Psychology Handbook*. Ed. by J. M. Doris. Oxford University Press.

Markman, K. D., I. Gavanski, S. J. Sherman, and M. N. McMullen (1993). "The mental simulation of better and worse possible worlds". In: *Journal of Experimental Social Psychology* 29, pp. 87–109.

McCloy, R. and R. M. J. Byrne (2000). "Counterfactual thinking about controllable events". In: *Memory and Cognition* 28, pp. 1071–1078.

McCormack, T., C. Frosch, and P. Burns (2011). "The Relationship between Children's Causal and Counterfactual Judgements". In: *Understanding Counterfactuals, Understanding Causation*. Ed. by C. Hoerl, T. McCormack, and S. R. Beck. Oxford University Press: Oxford, UK.

McIntyre, A. (2004). "Doctrine of Double Effect". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Fall 2011. `http://plato.stanford.edu/archives/`

fall2011/entries/double-effect/. Center for the Study of Language and Information, Stanford University.

McLaren, B. M. (2003). "Extensionally Defining Principles and Cases in Ethics: an AI Model". In: *Artificial Intelligence Journal* 150, pp. 145–181.

McLaren, B. M. and K. D. Ashley (1995). "Case-Based Comparative Evaluation in TruthTeller". In: *Procs. 17th Annual Conference of the Cognitive Science Society*.

Migliore, S., G. Curcio, F. Mancini, and S. F. Cappa (2014). "Counterfactual thinking in moral judgment: an experimental study". In: *Frontiers in Psychology* 5, p. 451.

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

Moor, J. H. (2011). "The Nature, Importance, and Difficulty of Machine Ethics". In: *Machine Ethics*. Ed. by M. Anderson and S. L. Anderson. Cambridge University Press: Cambridge, MA.

Muggleton, S. (1991). "Inductive Logic Programming". In: *New Generation Computing* 8(4), pp. 295–318.

Murakami, Y. (2004). "Utilitarian Deontic Logic". In: *Procs. 5th Advances in Modal Logic conference (AiML)*.

Nagel, T. (2014). "Listening to Reason". In: *The New York Review* 61(15), pp. 47–49.

National Society of Professional Engineers (NSPE) (1996). *The NSPE Ethics Reference Guide*. The National Society of Professional Engineers: Alexandria, VA.

Newman, J. O. (2006). "Quantifying the standard of proof beyond a reasonable doubt: a comment on three comments". In: *Law, Probability and Risk* 5(3-4), pp. 267–269.

Niemelä, I. and P. Simons (1997). "Smodels: an implementation of the stable model and well-founded semantics for normal LP". In: *Procs. 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Vol. 1265. LNAI.

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

Pearl, J. (2009). *Causality: Models, Reasoning and Inference*. Cambridge University Press: Cambridge, MA.

Peirce, C. S. (1932). *Collected Papers of Charles Sanders Peirce. Volume II. Elements of Logic*. Ed. by C. Hartshorne and P. Weiss. Harvard University Press: Cambridge, MA.

Pereira, L. M. (2015). *Software sans Emotions but with Ethical Discernment*. To appear in *Morality and Emotion: (Un)conscious Journey to Being*. Available from: http://centria.di.fct.unl.pt/~lmp/publications/online-papers/Software%20sans%20Emotions.pdf.

Pereira, L. M. and T. A. Han (2009). "Evolution Prospection". In: *Procs. 1st KES International Symposium on Intelligent Decision Technologies (IDT)*. Vol. 199, pp. 139–150.

Pereira, L. M. and G. Lopes (2009). "Prospective logic agents". In: *International Journal of Reasoning-based Intelligent Systems* 1(3/4), pp. 200–208.

Pereira, L. M. and A. Saptawijaya (2015). "Bridging Two Realms of Machine Ethics". In: *Rethinking Machine Ethics in the Age of Ubiquitous Technology*. Ed. by J. B. White and R. Searle. IGI Global: Hershey, PA.

Pereira, L. M., J. N. Aparício, and J. J. Alferes (1991a). "Counterfactual Reasoning Based on Revising Assumptions". In: *Procs. International Symposium on Logic Programming (ILPS 1991)*. MIT Press, pp. 566–577.

Pereira, L. M., J. N. Aparício, and J. J. Alferes (1991b). "Hypothetical Reasoning with Well Founded Semantics". In: *Procs. 3rd Scandinavian Conference on Artificial Intelligence*. IOS Press.

Pereira, L. M., C. V. Damásio, and J. J. Alferes (1993a). "Debugging by Diagnosing Assumptions". In: *Automatic Algorithmic Debugging*. Vol. 749. LNCS. Springer, pp. 58–74.

Pereira, L. M., C. V. Damásio, and J. J. Alferes (1993b). "Diagnosis and Debugging as Contradiction Removal in Logic Programs". In: *Progress in Artificial Intelligence*. Vol. 727. LNAI. Springer, pp. 183–197.

Pereira, L. M., P. Dell'Acqua, A. M. Pinto, and G. Lopes (2013). "Inspecting and Preferring Abductive Models". In: *The Handbook on Reasoning-Based Intelligent Systems*. Ed. by K. Nakamatsu and L. C. Jain. World Scientific Publishers, pp. 243–274.

Pereira, L. M., E.-A. Dietz, and S. Hölldobler (2014). "Contextual Abductive Reasoning with Side-Effects". In: *Theory and Practice of Logic Programming* 14(4-5), pp. 633–648.

Pereira, L. M., E.-A. Dietz, and S. Hölldobler (2015). *An Abductive Counterfactual Reasoning Approach in Logic Programming*. Available from `http://goo.gl/bx0mIZ`.

Pinheiro, F. L., J. M. Pacheco, and F. C. Santos (2012). "From Local to Global Dilemmas in Social Networks". In: *PLoS ONE* 7(2), e32114. doi:10.1371/journal.pone.0032114.

Poole, D. (1997). "The Independent Choice Logic for modelling multiple agents under uncertainty". In: *Artificial Intelligence* 94(1-2), pp. 7–56.

Poole, D. L. (1988). "A Logical framework for Default Reasoning". In: *Artificial Intelligence* 36(1), pp. 27–47.

Powers, T. M. (2006). "Prospects for a Kantian machine". In: *IEEE Intelligent Systems* 21(4), pp. 46–51.

Przymusinska, H. and T. C. Przymusinski (1990). "Semantic Issues in Deductive Databases and Logic Programs". In: *Formal Techniques in Artificial Intelligence: A Sourcebook*. North-Holand, pp. 321–367.

Przymusinski, T. C. (1989a). "Every Logic Program Has a Natural Stratification and an Iterated Least Fixed Point Model". In: *Procs. 8th ACM Symposium on Principles Of Database Systems (PODS)*, pp. 11–21.

Przymusinski, T. C. (1989b). "Three-valued non-monotonics formalisms and logic programming". In: *Procs. 1st International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Raedt, L. D., A. Kimmig, and H. Toivonen (2007). "ProbLog: a probabilistic Prolog and its application in link discovery". In: *Procs. 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Rawls, J. (1971). *A Theory of Justice*. Harvard University Press: Cambridge, MA.

Ray, O., A. Antoniades, A. Kakas, and I. Demetriades (2006). "Abductive logic programming in the clinical management of HIV/AIDS". In: *Proc. 17th. European Conference on Artificial Intelligence*. IOS Press.

Reiter, R. (1980). "A Logic for Default Reasoning". In: *Artificial Intelligence* 13, pp. 81–132.

Riguzzi, F. and T. Swift (2011). "The PITA system: Tabling and answer subsumption for reasoning under uncertainty". In: *Theory and Practice of Logic Programming* 11(4-5), pp. 433–449.

Riguzzi, F. and T. Swift (2014). *Probabilistic Logic Programming under the Distribution Semantics*. To appear in *Festschrift in honor of David S. Warren*. Available from `http://coherentknowledge.com/wp-content/uploads/2013/05/plp-festschrift-paper-TS+FR.pdf`.

Roese, N. J. (1997). "Counterfactual Thinking". In: *Psychological Bulletin* 121(1), pp. 133–148.

Ross, W. D. (1930). *The Right and the Good*. Oxford University Press: Oxford, UK.

Saha, D. (2006). "Incremental Evaluation of Tabled Logic Programs". PhD thesis. SUNY Stony Brook.

Saha, D. and C. R. Ramakrishnan (2005). "Incremental and demand-driven points-to analysis using logic programming". In: *Procs. 7th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP)*. ACM, pp. 117–128.

Saha, D. and C. R. Ramakrishnan (2006). "A Local Algorithm for Incremental Evaluation of Tabled Logic Programs". In: *Procs. 22nd International Conference on Logic Programming (ICLP)*. Vol. 4079. LNCS. Springer, pp. 56–71.

Saptawijaya, A. and L. M. Pereira (2013a). "Program Updating by Incremental and Answer Subsumption Tabling". In: *Procs. 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Vol. 8148. LNCS. Springer, pp. 479–484.

Saptawijaya, A. and L. M. Pereira (2013b). "Towards Practical Tabled Abduction Usable in Decision Making". In: *Procs. 5th KES International Conference on Intelligent Decision Technologies (IDT)*. Frontiers of Artificial Intelligence and Applications (FAIA). IOS Press.

Saptawijaya, A. and L. M. Pereira (2014). "Joint Tabling of Logic Program Abductions and Updates (Technical Communication of ICLP 2014)". In: *Theory and Practice of Logic Programming, Online Supplement* 14(4-5). Available from `http://arxiv.org/abs/1405.2058`.

Saptawijaya, A. and L. M. Pereira (2015). "TABDUAL: a Tabled Abduction System for Logic Programs". In: *IfCoLog Journal of Logics and their Applications* 2(1), pp. 69–123.

Sato, T. (1995). "A statistical learning method for logic programs with distribution semantics". In: *Procs. 12th International Conference on Logic Programming (ICLP)*.

Scanlon, T. M. (1998). *What We Owe to Each Other*. Harvard University Press: Cambridge, MA.

Scanlon, T. M. (2008). *Moral Dimensions: Permissibility, Meaning, Blame*. Harvard University Press: Cambridge, MA.

Scanlon, T. M. (2014). *Being Realistic about Reasons*. Oxford University Press: Oxford, UK.

Segbroeck, S. V., S. D. Jong, A. Nowé, F. C. Santos, and T. Lenaerts (2010). "Learning to coordinate in complex networks". In: *Adaptive Behavior* 18(5), pp. 416–427.

Shane, S. (2012). "The Moral Case for Drones". In: *Sunday Review, The New York Times* (14 July).

Singer, P. W. (2009). *Wired for War*. The Penguin Press: East Rutherford, NJ.

Sinnott-Armstrong, W. (2003). "Consequentialism". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Spring 2014. `http://plato.stanford.edu/archives/spr2014/entries/consequentialism/`. Center for the Study of Language and Information, Stanford University.

Stanovich, K. E. (2011). *Rationality and the Reflective Mind*. Oxford University Press: Oxford, UK.

Swift, T. (1999). "Tabling for non-monotonic programming". In: *Annals of Mathematics and Artificial Intelligence* 25(3-4), pp. 201–240.

Swift, T. (2014). "Incremental Tabling in Support of Knowledge Representation and Reasoning". In: *Theory and Practice of Logic Programming* 14(4-5), pp. 553–567.

Swift, T. and D. S. Warren (2010). "Tabling with Answer Subsumption: Implementation, Applications and Performance". In: *JELIA 2010*. Vol. 6341. LNCS. Springer, pp. 300–312.

Swift, T. and D. S. Warren (2012). "XSB: Extending Prolog with Tabled Logic Programming". In: *Theory and Practice of Logic Programming* 12(1-2), pp. 157–187.

Swift, T., D. S. Warren, K. Sagonas, J. Freire, P. Rao, B. Cui, E. Johnson, L. de Castro, R. F. Marques, D. Saha, S. Dawson, and M. Kifer (2015). *The XSB System Version 3.6.x Volume 1: Programmer's Manual*.

Tetlock, P. E., P. S. Visser, R. Singh, M. Polifroni, A. Scott, S. B. Elson, P. Mazzocco, and P. Rescober (2007). "People as intuitive prosecutors: the impact of social-control goals on attributions of responsibility". In: *Journal of Experimental Social Psychology* 43, pp. 195–209.

The Economist (2012a). *March of the robots*. Technology Quarterly (pages 11-12).

The Economist (2012b). *Morals and the machine*. Main Front Cover and Leaders (page 13).

The Future of Life Institute (2015a). *International Grant Competition for Robust and Beneficial AI*. `http://futureoflife.org/grants/large/initial`.

The Future of Life Institute (2015b). *Research Priorities for Robust and Beneficial Artificial Intelligence*. `http://futureoflife.org/static/data/documents/research_priorities.pdf`.

Thomson, J. J. (1985). "The trolley problem". In: *The Yale Law Journal* 279, pp. 1395–1415.

Tomasello, M. (2014). *A Natural History of Human Thinking*. Harvard University Press: Cambridge, MA.

Trappl, R. (2013). *Austrian Research Institute for AI (OFAI) Workshop on A Construction Manual for Robot's Ethical Systems: Requirements, Methods, Implementations*.

Trappl, R., ed. (2015). *A Construction Manual for Robots' Ethical Systems: Requirements, Methods, Implementations*. Cognitive Technologies (in press). Springer: Berlin.

Vennekens, J., S. Verbaeten, and M. Bruynooghe (2004). "Logic Programs with Annotated Disjunctions". In: *Procs. 20th International Conf. on Logic Programming (ICLP)*. Vol. 3132. LNCS. Springer.

Vennekens, J., M. Bruynooghe, and M. Denecker (2010). "Embracing events in causal modeling: Interventions and counterfactuals in CP-logic". In: *JELIA 2010*. Vol. 6341. LNCS. Springer, pp. 313–325.

Wallach, W. and C. Allen (2009). *Moral Machines: Teaching Robots Right from Wrong*. Oxford University Press: Oxford, UK.

Warren, D. (2013). "Interning ground terms in XSB". In: *Colloquium on Implementation of Constraint and Logic Programming Systems (CICLOPS 2013)*.

Weiner, B. (1995). *Judgments of Responsibility: A Foundation for a Theory of Social Conduct*. The Guilford Press: New York, NY.

Weiner, T. (2005). "New Model Army Soldier Rolls Closer to Battle". In: *The New York Times* (16 February).

White, J. B. and R. Searle, eds. (2015). *Rethinking Machine Ethics in the Age of Ubiquitous Technology*. IGI Global: Hershey, PA.

Woodward, J. (2011). "Psychological Studies of Causal and Counterfactual Reasoning". In: *Understanding Counterfactuals, Understanding Causation*. Ed. by C. Hoerl, T. McCormack, and S. R. Beck. Oxford University Press: Oxford, UK.