Abduction and Beyond in Logic Programming with Application to Morality

Luís Moniz Pereira
Ari Saptawijaya*
NOVA Laboratory for Computer Science and Informatics
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal
lmp@fct.unl.pt, ar.saptawijaya@campus.fct.unl.pt

Abstract

In this paper we emphasize two different aspects of abduction in Logic Programming (LP): (1) the engineering of LP abduction systems, and (2) application of LP abduction, complemented with other non-monotonic features, to model morality issues. For the LP engineering part, we present an implemented tabled abduction technique in order to reuse priorly obtained (and tabled) abductive solutions, from one abductive context to another. Aiming at the interplay between LP abduction and other LP non-monotonic reasoning, this tabled abduction technique is combined with our own-developed LP updating mechanism – the latter also employs tabling mechanisms, notably incremental tabling of XSB Prolog. The first contribution of this paper is therefore a survey of our tabled abduction and updating techniques, plus further development of our preliminary approach to combine these two techniques.

The second contribution of the paper pertains to the application part. We formulate a LP-based counterfactual reasoning, based on Pearl's structural theory, via the aforementioned unified approach of our LP abduction and updating. The formulation of counterfactuals allows us to subsequently demonstrate its applications to model moral permissibility, according to the Doctrines of Double and Triple Effect, and to provide its justification. The applications are shown through classic moral examples from the literature, and tested in our prototype, Qualm, an implementation reifying the presented unified approach.

We thank the referees for their constructive comments and suggestions. Both authors acknowledge the support from FCT/MEC NOVA LINCS PEst UID/CEC/04516/2013. Ari Saptawijaya acknowledges the support from FCT/MEC grant SFRH/BD/72795/2010. We thank Terrance Swift and David Warren for their expert advice in dealing with implementation issues in XSB, and Emmanuelle-Anna Dietz for fruitful discussions on counterfactuals.

*Affiliated also with the Faculty of Computer Science at Universitas Indonesia, Depok, Indonesia.

1 Introduction

Abduction has been well studied in Logic Programming (LP) [11, 26, 25, 13, 17, 4], establishing theoretical results that support the implementation of LP abduction systems. LP abduction has been applied in a variety of areas, such as in diagnosis [18], planning [15], scheduling [28], reasoning of rational agents and decision making [31, 42], knowledge assimilation [27], natural language understanding [5], security protocols verification [1], and systems biology [50]. Such applications demonstrate the potential of LP abduction and motivates further advancements of LP abduction systems to prepare them for even more challenging applications.

In this paper we report two results of our recent research in LP abduction concerning: (1) the engineering of LP abduction systems, and (2) application of LP abduction, complemented with other non-monotonic reasoning, to modeling a number of issues in morality.

Engineering of LP abduction systems We address two challenges in engineering a LP abduction system. For one, in abduction, finding some best explanations (i.e. adequate abductive solutions) to an observed evidence, or finding assumptions that can justify a goal, can be costly. 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. In LP, absent of abduction, goal solution reuse is commonly addressed by employing a tabling mechanism [62]. Therefore, tabling appears to be conceptually suitable for abduction, so as to reuse abductive solutions. In practice, abductive solutions reuse is not immediately amenable to tabling, because such solutions go together with an abductive context.

We conceptualize in [56] a *tabled abduction* technique, to benefit from LP tabling mechanisms in contextual abduction, i.e., to reuse priorly obtained (and tabled) abductive solutions, from one abductive context to another. The technique is underpinned by the theory of Abdual [4], for computing abduction in LP over Well-Founded Semantics [66]. The tabled abduction technique is implemented in XSB Prolog [64], resulting in a LP abduction system Tabdual. It concretely realizes the abstract theory of Abdual, while also taking care of pragmatic issues, due to the presence of tabled abduction, to foster the practicality of Tabdual.

Another challenge we address is the interplay between LP abduction and other LP non-monotonic reasoning. It is apparent that when it comes to applications, notably in reasoning of rational agents and decision making, abduction needs to be enriched with other features. For instance, such applications are susceptible to knowledge updates and changes, due to incomplete information of the world or the evolution of the agents themselves. In this case, abduction systems need to be

complemented with LP updating feature.

Aiming at that goal, we propound yet another use of tabling mechanisms, notably the so-called *incremental tabling*, for LP updating. Incremental tabling, also available in XSB Prolog, is an advanced tabling feature that ensures the consistency of answers in a table with all dynamic information 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. This incremental table maintenance is achieved by automatically propagating assertion or retraction of information bottom-up; such propagation is typically triggered by the invocation of a given goal. In [55, 54], 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 [3] and its subsequent development, Evolving Logic Programs (Evolp) [2].

In [57] we preliminarily bring LP abduction and LP updating into a unified approach based on the aforementioned two implementation techniques: Tabdual and Evolp/r, respectively. It aims at the use of joint tabling of both features, so as to benefit from reusing the abductive solutions obtained in one context for another, while also allowing top-down goal-driven incremental tabling of updates by upwards propagation. The approach in [57] borrows the concept of hypothesis generation [42]. While this concept permits generating only abductive explanations relevant for the problem at hand, the preliminary unified approach [57] limits the benefit of tabled abduction. In this paper we further develop this unified approach by leaving out the hypotheses generation concept in order to uphold the idea and the benefit of tabled abduction. The unified approach is implemented and applied to formulate counterfactual reasoning, which in turn is used to address morality issues, as we detail next.

Applications of LP abduction to Morality Computational morality (also known as machine ethics, machine morality, artificial morality and computational ethics) is a burgeoning field of inquiry that emerges from the need of imbuing autonomous agents with the capacity of moral decision-making. It has particularly attracted interest from the artificial intelligence community and has brought together perspectives from various fields, amongst them: philosophy, cognitive science, neuroscience and primatology. The overall result of this interdisciplinary research is therefore not only important for equipping agents with the capacity of making moral judgments, but also for helping us better understand morality, through the creation and testing of computational models of ethical theories.

The potential of LP to computational morality has been reported in [30, 45,

21]. In [58] we summarize our work emphasizing the application of LP abduction and other LP-based reasoning for modeling the dynamics of knowledge and moral cognition of an individual agent, where we employ LP abduction jointly with other non-monotonic reasoning. The reader is also referred to [46] which summarizes: (1) our study with other co-authors on collective morals and the emergence, in a population, of evolutionarily stable moral norms, of fair and just cooperation, in the presence of cognitive aspects, such as intention recognition, commitment, and mutual tolerance through apology; and (2) our view on bridging individual and collective morality, as they are necessarily intertwined.

In the second part of the paper we report the continuation of our prior work for modeling the dynamics of knowledge and moral cognition of an individual agent, now benefiting from the integration of Tabdual and Evolp/r discussed earlier. It is important to note that the application we show in this paper neither aims at defending any considered moral principles nor resolving the dilemmas appearing in the examples, as even philosophers naturally are split over opinions on them. Instead, its purpose is to show that our unified approach of LP abduction and updating (with its implementation) are capable and appropriate for expressing viewpoints on morality issues discussed herein, in accordance with the results and the theory argued in the literature.

We start by applying LP abduction and updating to model counterfactual reasoning, as people typically reason about what they should or should not have done when they examine decisions in moral situations [36, 34, 52, 38]. It is therefore natural for them to engage counterfactual thoughts in such settings.

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 [8]: *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".

Counterfactuals have been widely studied in philosophy [32, 9, 20], psychology [35, 51, 36, 8, 14, 38], as well as from the computational viewpoint [19, 41, 6, 40, 67]. Research in LP to model counterfactual reasoning is nevertheless still limited. In [41], counterfactuals, based on Lewis's counterfactuals [32], are evaluated using contradiction removal semantics of LP. 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. In [6], Probabilistic LP language P-log with Stable Model Semantics is employed to encode Pearl's probabilistic causal model [40] of counterfactual reasoning. In [67], Pearl's approach is encoded

using a different Probabilistic LP, viz., CP-logic. None of these LP-based approaches involves abduction in modeling counterfactuals.

In this paper we specifically adopt Pearl's approach [40], abstaining from probabilities, but resorting to LP abduction and updating. LP abduction is employed for providing background conditions from observations made or evidences given, whereas defeasible LP rules allow adjusting the current model via hypothetical updates of intervention. Therefore, our first application is to formulate an implemented evaluation procedure for counterfactuals via LP abduction and updating. The implementation is based on our unified approach of Tabdual and Evolp/r, and named Qualm (its ongoing development is available from `https://github.com/merah-putih/qualm`). Qualm is appropriately used for applications concerned with the reasoning of agents, in particular those involving counterfactuals when probabilities are not known or needed. We specifically look into its challenging applications in moral reasoning (for our related work on probabilistic moral reasoning, cf. [21]).

We apply counterfactuals (by resorting to its LP formulation), together with a combination of LP abduction and updating, to model morality issues. Counterfactuals are engaged to examine moral permissibility of an action by distinguishing whether an effect of an action is a cause for achieving a morally dilemmatic goal or merely a side-effect of that action. The distinction is essential for establishing moral permissibility from the viewpoints of the Doctrines of Double Effect [37] and of Triple Effect [29], as scrutinized herein through several classic moral examples from the literature. We also apply counterfactuals to address a form of moral justification. Through examples, we show how compound counterfactuals may provide moral justification for what was done due to lack of the current knowledge. We also touch upon Scanlon's contractualism [60] as a reference for employing counterfactuals to provide a justified, but defeasible, exception to permissibility of actions.

The paper is organized as follows. Section 2 reviews necessary background on abduction in LP. We discuss the LP engineering part by presenting the concept of tabled abduction, LP updating with incremental tabling, and the interplay between them in Section 3. The application part is subsequently elaborated in the next two sections: the formulation of counterfactuals with LP abduction and updating is detailed in Section 4, whereas Section 5 presents applications of counterfactuals to model morality issues. The reader may skip the technical LP engineering part (Section 3) without loss of understanding of the application part. We conclude in Section 6 with some remarks on related issues that may trigger future work.

2 Abduction in Logic Programming

We start by recapping basic notation in LP and review how abduction is expressed and computed in LP.

A *literal* is either an atom B or its default negation *not* B, named positive and negative literals, respectively. They are negation complements to each other. The atoms *true* and *false* are true and false, respectively, in every interpretation. A *logic program* is a set of rules of the form $H \leftarrow B$, naturally read as "H if B", where its *head* H is an atom and its (finite) *body* B is a sequence of literals. When B is empty (equal to *true*), the rule is called a *fact* and simply written H. A rule in the form of a denial, i.e., with *false* as head, is an *integrity constraint* (IC).

In LP, an abductive hypothesis (*abducible*) is a 2-valued positive literal Ab or its negation complement Ab^* (denotes *not* Ab), whose truth value is not initially assumed, and it does not appear in the head of a rule. An *abductive logic program* is one allowing abducibles in the body of rules. An observation O is a set of literals, analogous to a query in LP; we use the usual LP notation `?-` to denote a query.

Abduction in LP can be accomplished by a top-down query-oriented procedure for finding a query solution, i.e., a *consistent* abductive solution, by need. The solution's abducibles are leaves in its procedural query-rooted call-graph, where 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.

Example 1. A library in a city is close with two possible explanations: it is close in the weekend, or when it is not weekend, there are no librarians working. These days, librarians are often absent from their work because they participate in a strike. On the other hand, a museum in that city is only close when there is a special visit by important guests. This example can be expressed by a logic program below (*weekend*, *strike*, and *special_visit* are abducible atoms):

$$close\_library \leftarrow weekend. \qquad close\_library \leftarrow weekend^*, absent.$$
$$absent \leftarrow strike. \qquad close\_museum \leftarrow special\_visit.$$

Consider the query `?-` *close_library*. It induces the call-graph with *close_library* as its root and, through procedure calls, it ends with two leaves: one leaf containing abducible *weekend* as an abductive solution, and the other containing abducibles $[weekend^*, strike]$ as another solution.

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. In Example 1,

rule *close_museum* ← *special_visit* is not relevant to answer the aforementioned query, and thus is not involved in the call-graph.

The 3-valued Well-Founded Semantics or WFS [66] enjoys this relevancy property [12], 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, e.g. abducible *special_visit* is indifferent to query `?-` *close_library*. Due to its relevancy property, the approaches in this paper are based on WFS. Moreover, the XSB Prolog system [64], wherein all the prototypes of our research are implemented, computes the WFS.

3 Tabling in Contextual Abduction and LP Updating

The progress of LP has been promoting new techniques for engineering LP-based systems, most notably tabling mechanisms [62]. Tabling affords solutions reuse, rather than recomputing them, by keeping in tables subgoals and their answers obtained by query evaluation. It is supported, to a different extent, by a number of Prolog systems.

In abduction, finding explanations to an observed evidence, or finding assumptions that can justify a goal, can be costly. We discuss in Section 3.1, that abduction may benefit from tabling mechanisms by reusing priorly obtained abductive solutions from one abductive context to another. We subsequently discuss a unified approach of LP abduction and tabling via their joint tabling, in Section 3.2.

3.1 Tabling in Contextual Abduction

Motivation and Idea Example 2 illustrates the motivation and the idea of tabled abduction.

Example 2. A plane may be hit by lightning, when it flies in a stormy area and at a low altitude. A case where it flies into a stormy area is when it enters cumulonimbus cloud (a dense towering cloud associated with thunderstorms). Being hit by lightning in such a stormy area may crash the plane. This example is expressed in the program below (*cumulonimbus* and *low_altitude* are abducible atoms):

$$storm \leftarrow cumulonimbus. \qquad hit\_lightning \leftarrow storm, low\_altitude.$$
$$crash \leftarrow hit\_lightning, storm.$$

Suppose three queries are invoked, asking for individual explanations of *storm*, *hit_lightning*, and *crash*, in that order. The first query, `?-` *storm*, is satisfied simply by having [*cumulonimbus*] as the abductive solution for *storm*, and tabling it.

Executing the second query, `?-` *hit_lightning*, amounts to satisfying the two subgoals in its body, i.e., abducing *low_altitude* followed by invoking *storm*. Since *storm* has previously been invoked, we can benefit from reusing its solution, instead of recomputing, given that the solution was tabled. That is, query `?-` *hit_lightning* can be solved by extending the current ongoing abductive context [*low_altitude*] of subgoal *storm* with the already tabled abductive solution [*cumulonimbus*] of *storm*, yielding the abductive solution [*cumulonimbus*, *low_altitude*].

The final query `?-` *crash* can be solved similarly. Invoking the first subgoal *hit_lightning* results in the priorly registered abductive solution: [*cumulonimbus*, *low_altitude*], which becomes the current abductive context of the second subgoal *storm*. Since [*cumulonimbus*, *low_altitude*] subsumes the previously obtained (and tabled) abductive solution [*cumulonimbus*] of *storm*, we can then safely take [*cumulonimbus*, *low_altitude*] as the abductive solution to query `?-` *crash*.

Example 2 shows how a tabled abductive solution [*cumulonimbus*], the abductive solution of the first query `?-` *storm*, can be reused from one abductive context of *storm* (viz., [*low_altitude*] in the second query, `?-` *hit_lightning*) to another context (viz., [*cumulonimbus*, *low_altitude*] in the third query, `?-` *crash*). In practice a repeatedly called rule, like *storm*, may have a body comprising many subgoals, causing potentially expensive recomputation of its abductive solutions, if they have not been tabled.

Contextual Abduction with Tabling Tabled abduction with its prototype Tabdual consists of a program transformation that provides self-sufficient program transforms, which can be directly run to enact abduction by means of Tabdual's library of reserved predicates. We describe below how the idea in Example 2 is conceptually realized by contextual abduction with tabling in Tabdual. The reader is referred to [59] for a more formal and technical treatment of the transformation.

Example 2 indicates two key ingredients of tabled abduction in Tabdual:

1. *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.

2. *Tabled predicates*, which table the abductive solutions for predicates defined in the program, such that they can be reused from one abductive context to another.

Example 3 shows how these two ingredients figure in the program transform of Example 2.

Example 3. Consider rule $storm \leftarrow cumulonimbus$ of Example 2. A new predicate $storm_{ab}$ is introduced to table the abductive solution of $storm$,[1] defined as:

$$storm_{ab}([cumulonimbus]).$$

It is a simple tabled fact, as the rule of *storm* is defined solely by the abducible *cumulonimbus*, without any other literals in its body.

We defined similar tabled predicates $hit\_lightning_{ab}$ and $crash_{ab}$ for the other two rules (of *hit_lightning* and *crash*, respectively):

$$hit\_lightning_{ab}(E) \leftarrow storm([low\_altitude], E). \quad (1)$$

$$crash_{ab}(E_2) \leftarrow hit\_lightning([\,], E_1), storm(E_1, E_2). \quad (2)$$

Predicate $hit\_lightning_{ab}(E)$ is a predicate that tables, in E, the abductive solution of *hit_lightning*. Its definition follows from the original definition of *hit_lightning*. Two extra (and new) parameters, that serve as *input* and *output contexts*, are added to the subgoal *storm*. Notice how *low_altitude*, the abducible appearing in the body of the original rule of *hit_lightning*, becomes the input abductive context of *storm*. This subgoal *storm* requires a rule of $storm(I, O)$, defined in rule (3) below, that reuses the solution tabled in $storm_{ab}$ to produce the output abductive context O from its input context I.

Predicate $crash_{ab}$ tables the abductive solution for *crash* in its argument E_2. The rule expresses that the tabled abductive solution E_2 of $crash_{ab}$ is obtained by relaying the ongoing abductive solution stored in context E_1 from subgoal *hit_lightning* to subgoal *storm* in the body, given the empty input abductive context (i.e., [ ]) of *hit_lightning* (because there is no abducible by itself in the body of the original rule of *crash*).

Now that we have tabled predicates, the remaining rules to define are those that reuse the tabled solutions. That is, we define the rule of $storm(I, O)$ that reuses the solution tabled in $storm_{ab}$, and produce the output abductive context O from its input context I (the other rules $hit\_lightning(I, O)$ and $crash(I, O)$ are defined similarly):

$$storm(I, O) \leftarrow storm_{ab}(E), produce\_context(O, I, E). \quad (3)$$

This rule expresses that the output abductive solution O of *storm* is obtained from the solution entry E of $storm_{ab}$ and the given input context I of *storm*, via the Tabdual system predicate $produce\_context(O, I, E)$. This system predicate concerns itself with: whether E is already contained in I and, if not, whether there

[1] In XSB Prolog, this tabling effect is achieved by declaring predicate $storm_{ab}$ as a tabled predicate.

are any abducibles from E, consistent with I, that can be added to produce O. For example, *produce_context* in rule (1) adds the tabled solution *cumulonimbus* to the input abductive context $[low\_altitude]$ of *storm*, thus producing the output $[cumulonimbus, low\_altitude]$. On the other hand, in rule (2), the tabled solution *cumulonimbus* of *storm* is already contained in the input abductive context $E_1 = [cumulonimbus, low\_altitude]$, yielding the latter as the output $E_2 = E_1$.

Note that if E is inconsistent with I then the specific entry E cannot be reused with I, *produce_context* fails and another entry E is sought. For instance, suppose *storm* is invoked with the input abductive context $[cumulonimbus^*]$. This goal $storm([cumulonimbus^*], O)$ will fail, as *produce_context* ends up with an inconsistent solution containing *cumulonimbus* and $cumulonimbus^*$. In summary, *produce_context* should guarantee that it produces a *consistent* output context O from I and E that encompasses both.

Abduction under Negative Goals We have seen in Example 3 abduction performed under positive goals via abductive contexts and tabling. A common approach for abducing under negative goals is by first computing all abductive solutions of its corresponding positive goal, and then having to negate their disjunction. Tabdual avoids such computation of all abductive solutions by employing the *dual transformation* [4] in contextual abduction.

The dual transformation is based on the idea of making negative goals 'positive' literals. It thus permits obtaining one abductive solution at a time, just as if we treat abduction under positive goals. The dual transformation defines for each atom A and its set of rules R in a program P, a set of dual rules whose head not_A is true if and only if A is false by R in the employed semantics of P. Note that, instead of having a negative goal *not* A as the rules' head, we use its corresponding 'positive' literal, not_A. Example 4 illustrates the main idea of how the dual transformation is employed in Tabdual.

Example 4. Recall Example 1 and the two rules of *close_library*:

$$close\_library \leftarrow weekend. \qquad close\_library \leftarrow weekend^*, absent.$$

The dual transformation will create a set of dual rules for *close_library*. These dual rules falsify *close_library* with respect to its two rules. That is, the rule of *not_close_library* is defined by falsifying both the first rule *and* the second rule, denoted below by predicate $close\_library^{*1}$ and $close\_library^{*2}$, respectively:

$$not\_close\_library(C_0, C_2) \leftarrow close\_library^{*1}(C_0, C_1),\ close\_library^{*2}(C_1, C_2).$$

Note the input and output abductive context parameters that figure in the rule:

C_0 and C_2, in the head, and similarly in each subgoal of the rule's body, where intermediate context C_1 relays the ongoing abductive solution from $close\_library^{*1}$ to $close\_library^{*2}$.

Next, we define how the first and the second rules of *close_library* are falsified, i.e., rules for $close\_library^{*1}$ and $close\_library^{*2}$. These rules are naturally defined by falsifying the body of *close_library*'s first rule and second rule, respectively. In case of $close\_library^{*1}$: the first rule of *close_library* is falsified only by abducing the negation of *weekend*, viz., $weekend^*$. Therefore, we have:

$$close\_library^{*1}(I, O) \leftarrow weekend^*(I, O).$$

Notice that $weekend^*$ is abduced by invoking the subgoal $weekend^*(I, O)$. This subgoal is defined as follows (other abducibles, both positive and negative ones, transform similarly):

$$weekend^*(I, O) \leftarrow insert\_abducible(weekend^*, I, O). \tag{4}$$

where $insert\_abducible(A, I, O)$ is a Tabdual system predicate that inserts abducible $weekend^*$ into input context I, resulting in output context O. As in the predicate *produce_context*, it also maintains the consistency of the context, failing if inserting A results in an inconsistent one.

In case of $close\_library^{*2}$: the second rule of *close_library* is falsified by alternatively failing one subgoal in its body at a time, i.e. by negating $weekend^*$ or, alternatively by negating *absent*.

$$close\_library^{*2}(I, O) \leftarrow weekend(I, O).$$
$$close\_library^{*2}(I, O) \leftarrow not\_absent(I, O).$$

where $weekend(I, O)$ is similarly defined as in the above rule of $weekend^*(I, O)$, and $not\_absent(I, O)$ is defined by the dual transformation for *absent*.

Consider now query `?-` $not\_close\_library([\,], O)$. Provided the dual rules for *close_library* and *absent*, we obtain a single solution $[weekend^*, strike^*]$. Note that *insert_abducible* rules out $[weekend^*, weekend]$ as a solution, due to its inconsistency.

Abduction with Integrity Constraints

Example 5. Recall Example 1 plus new information: librarians may also be free from working in case the library is being renovated, expressed as:

$$absent \leftarrow renov.$$

Query `?-` $close\_library([\,], T)$ now returns an additional solution, viz. $[weekend^*, renov]$. Let us further suppose that the municipal authority permits any renovations to take place only on weekends, expressed as an IC below:

$$false \leftarrow weekend^*, renov.$$

The queries should now respect the IC, too. That is, it should be conjoined with *not_false* to ensure that all integrity constraints are satisfied:

$$\texttt{?-}\ close\_library([\,], T), not\_false(T, O).$$

where the dual transformation provides the definition for not_false. Note that the abductive solution for *close_library* is further constrained by passing it to the subsequent subgoal *not_false* for confirmation, via the intermediate context T. Using the Tabdual approach, we can easily check that $[weekend^*, renov]$ is now ruled out from the solutions.

Tabdual also takes care of abduction over non-ground programs and programs with loops. Abduction over programs with loops sometimes occur in real applications, e.g., in psychiatric diagnosis, cf. [18] that deals with non-stratified negation: "The patient has an Adjustment Disorder if he does not have Alzheimer's Dementia, and has Alzheimer's Dementia if the patient does not have an Adjustment Disorder", which corresponds to rules $adjustment\_disorder \leftarrow not\ alzheimer\_dementia$ and $alzheimer\_dementia \leftarrow not\ adjustment\_disorder$. Tabdual also benefits from other XSB's features to foster its practicality, e.g., constructing dual rules by need only. The reader is referred to [59] for details of implementation aspects.

3.2 Combining LP Abduction and Updating

In addition to abduction, an agent may learn new knowledge from the external world or update itself internally on its own decision in order to pursue its present goal. It is therefore natural to accommodate abduction systems with knowledge updating. In Section 3.2.1 we summarize an implemented approach for LP updating, called Evolp/r, that employs incremental tabling [53], an advanced tabling feature of XSB Prolog that lends itself to dynamic environments and evolving systems. Having the two tabling-based approaches for different purposes, viz., Tabdual and Evolp/r, we combine them in a unified approach, as discussed in Section 3.2.2.

3.2.1 Evolp/r for LP Updating

Evolp/r is theoretically underpinned by Evolving Logic Programs (Evolp) [2]; the latter is itself based on Dynamic Logic Program [3]. Though its syntax and semantics are based on Evolp, they differ in several respects, which characterize

Evolp/r, as per below. The reader is referred to [54] for a more detailed theoretical basis of Evolp/r.

Restriction to Fluent Updates While Evolp allows full-blown rule updates, Evolp/r restricts updates to state-dependent literals only (commonly known as *fluents*), i.e., literals whose truth value may change from one state to another. Syntactically, every fluent F is accompanied by its fluent complement $\sim F$. Program updates are enacted by asserting F, whereas its retraction is realized by asserting its complement $\sim F$ at a later state. It thus admits non-monotonicity of a fluent, as the latter update supervenes the former.

Though updates in Evolp/r are restricted to fluents only, it nevertheless still permits rule updates by introducing a rule name fluent that uniquely identifies the rule for which it is introduced. Such a rule name fluent is placed in the body of a rule to turn the rule on and off, cf. [48]; this being achieved by asserting or retracting that specific fluent. For instance, while Evolp allows asserting rule $absent \leftarrow renov$ in Example 5, Evolp/r does this by introducing a unique rule name fluent, say $rule_{(absent \leftarrow renov)}$, in the body of that rule (this is internally done by Evolp/r via a program transformation):

$$absent \leftarrow rule_{(absent \leftarrow renov)}, renov. \tag{5}$$

and asserting the rule name fluent $rule_{(absent \leftarrow renov)}$ at state T, making it true at T and thus turning the rule on. On the other hand, asserting $\sim rule_{(absent \leftarrow renov)}$ at a subsequent state $T' > T$, making the rule name fluent false later at T', and hence turning the rule off from then on.

Incremental Tabling of Fluents The first approach of Evolp/r [55] preliminarily exploits the combination of two tabling features in XSB Prolog: (1) *incremental tabling* [53], which ensures the consistency of answers in tables with all dynamic facts and rules upon which the tables depend; and (2) *answer subsumption* [63], which allows tables to retain only answers that subsume others wrt. some order relation.

Incremental tabling of fluents is employed in Evolp/r to automatically maintain the consistency of program states due to assertion and retraction of fluent literals, whether obtained as updated facts (wrt. extensional fluent) or concluded by rules (wrt. to intensional fluent). On the other hand, answer subsumption of fluent literals allows to address the frame problem, by automatically keeping track of only their latest assertion or retraction wrt. the state of a given query. Tabling both a fluent and its complement provides a convenient formulation of counterfactual reasoning (cf. Section 3.5 of [47]).

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 states (timestamps) when it holds true (*holds-time*) and when it is queried (*query-time*), resp. 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 time Ht_F fluent F is true wrt. a given query-time Qt. In order to return only the latest time Ht_F when F is true (wrt. Qt), $fluent/3$ is tabled using answer subsumption on its second argument. Predicate $fluent/3$ is then employed in Evolp/r system predicate $holds(F, Qt)$ to query whether F holds true at query-time Qt. It does so by looking for the entry for F in the table of $fluent(F, Ht_F, Qt)$, obtaining the latest time $Ht_F \leq Qt$, and guarantees that F is not supervened by its complement $\sim F$, i.e., $Ht_{\sim F} \leq Ht_F$, where $Ht_{\sim F}$ is obtained by invoking $fluent(\sim F, Ht_{\sim F}, Qt)$).

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.

The above issue is addressed in the second approach [54], 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 $fluent(F, Ht_F)$. Invoking $fluent(F, Ht_F)$ thus amounts to look for an entry in its table, if one exists; otherwise, it invokes dynamic definitions of fluent F, and returns a holds-time Ht_F (a state when fluent F is true). Since answer subsumption is left out, more than one instance of holds-time Ht_F of fluent F may be tabled.
The implementation details of Evolp/r are beyond the scope of this paper, but can be found in [54]. We only illustrate, in Example 6, how the state information, viz., holds-time, figures in a rule transform.

Example 6. Recall rule (5): $absent \leftarrow rule_{(absent \leftarrow renov)}, renov$. For the purpose of this example, assume that $renov$ is not an abducible, but instead a fluent. Its rule transform is shown below (abstracted away from implementation details):

$$absent(T) \leftarrow fluent(rule_{(absent \leftarrow renov)}, T_1),\ fluent(renov, T_2),\ latest([T_1, T_2], T).$$

where the reserved predicate *latest* determines a holds-time T of fluent *absent* in the head by which inertial fluent in its body (i.e., fluents $rule_{(absent \leftarrow renov)}$ and *renov*) holds the latest.

The main characteristics of the second approach are summarized below.

- Though more than one instance of holds-time Ht_F of fluent F may be tabled, this approach still permits avoiding top-down recursion or bottom-up iteration through the frame axiom. This is taken care of by the underlying tabling mechanism: look-up a collection of states a fluent literal is true in the table, pick-up the most recent one, and ensure that its complement, with a later state, does not exist in the table. More precisely, querying whether fluent F holds true at query-time Qt amounts to looking for entries for fluent F in the table, by invoking $fluent(F, Ht_F)$, in order to obtain the highest state $Ht_F \leq Qt$, and to guarantee that F is not supervened by its complement $\sim F$, by invoking $fluent(\sim F, Ht_{\sim F})$ and obtaining the highest state $Ht_{\sim F}$ such that $Ht_{\sim F} \leq Ht_F$.

- Propagation of fluent updates is controlled by initially keeping them pending in the database. On the initiative of an actual (top-goal) query, just those updates up to the actual query time are activated, via their incremental assertions, which in turn, automatically trigger system-level incremental bottom-up recomputation of other tabled fluent literals (viz., updating the state information Ht of these fluents). This updating technique thus allows high-level top-down deliberative reasoning about a query to be reconciled with autonomous low-level (via tabling) bottom-up world reactivity to ongoing updates.

- For facilitating the propagation of fluent complements, Evolp/r borrows the dual transformation from Tabdual. That is, the transformation derives rules for the dual negation fluent literal $\sim F$ from their corresponding positive rules of fluent F. It thus helps propagate this dual negation fluent literal incrementally, in order to establish whether the fluent or rather its complement is true at some state.

3.2.2 Combining Tabdual and Evolp/r

We have presented in the previous sections approaches for LP abduction (Tabdual) and updating (Evolp/r). Both approaches enjoy the benefit of tabling features, albeit their different usage. We now recap a unified approach to integrate Tabdual and Evolp/r for keeping the benefit of tabling in each individual approach.

Our initial attempt, as reported in [57], employs the concept of hypotheses generation from [42], where the notion of expectation is employed to express preconditions for enabling the assumption of an abducible. An abducible A can be assumed, or *considered*, if there is an expectation for it, and no expectation to the contrary:

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

Note that this concept consequently requires every rule with abducibles in its body to be preprocessed, by substituting abducible A with $consider(A)$. The need and the rule definitions for $expect(A)$ and $expect\_not(A)$ depend on the encoded problem.

While this concept generates only abductive explanations relevant for the problem at hand, the approach in [57] limits the benefit of tabled abduction. That is, instead of associating tabled abduction to a particular atom (e.g., $storm_{ab}$ in Example 3) for a subsequent reuse in another context, a single (and generic) tabled predicate $consider_{ab}$ is introduced for tabling only this single abducible A being considered. 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 will not depend on another $consider/1$.

Such a generic tabling by *consider_ab* is indeed unnecessary and not intended by tabled abduction. In this section we remedy the joint tabling approach of [57] in order to uphold the idea and the benefit of tabled abduction. As in [57], the joint tabling approach in combining Tabdual and Evolp/r naturally depends on two pieces of information carried from each of these techniques, viz. abductive contexts and the state when a fluent holds true, respectively. They keep the same function in the integration as in their respective individual approach. Example 7 shows how both entries figure in a remedied rule transform.

Example 7. Recall rule $hit\_lightning \leftarrow storm, low\_altitude$ in Example 2. Its rule transform is shown below (abstracted away from its rule name and other implementation details):

$$\begin{aligned} hit\_lightning_{ab}(E_2, T) \leftarrow\ & low\_altitude([\,], E_1, T_1), storm(E_1, E_2, T_2), \\ & latest([T_1, T_2], T). \end{aligned} \tag{6}$$

where E_2 is an abductive solution tabled by predicate $hit\_lightning_{ab}$, obtained by relaying the ongoing abductive solution in context E_1 from subgoal *low_altitude* to subgoal *storm* in the body, given the empty input abductive context (i.e., $[\,]$) of *low_altitude*. As in Example 6, the reserved predicate *latest* determines the state T of *hit_lightning* from *low_altitude* and *storm* that latest holds.

There are two points to note in the above remedied transformation. First, the abducible *low_altitude* in the body is now called explicitly (rather than immediately placed as the input abductive context of *storm*, cf. rule (1) in Example 3). This is to anticipate possible updates on this abducible. Such an abducible update may take place when one wants to commit to a preferred explanation and fix it in the program as a fact, a case that we show in Section 4 to fix an abduced background context of a counterfactual. Having an abducible as an explicit subgoal in the body thus facilitates bottom-up propagation of its updates (induced by incremental tabling in Evolp/r), due to the apparent dependency between the tabled predicate in the head (e.g., $hit\_lightning_{ab}$), and the abducible goal in the body (e.g., *low_altitude*). Second, by prioritizing abducible goals in the body (to occur before any other non-abducible goals), Tabdual's benefit, of reusing tabled abductive solutions from one context to another, is still obtained. In Example 7, calling abducible *low_altitude* with the empty input abductive context – solved using its definition similar to rule (4) – provides *storm* with an actual input abductive context $E_1 = [low\_altitude]$. It thus achieves the same effect as simply having $[low\_altitude]$ as the input context of *storm* (cf. rule (1) of Example 3). Using a similar transform as in rule (6), the rule transform for $crash_{ab}$ is given below:

$$crash_{ab}(E_1, T) \leftarrow hit\_lightning([\,], E_1, T_1), storm(E_1, E_2, T_2), latest([T_1, T_2], T).$$

The tabled solution in $hit\_lightning_{ab}$ (rule (6)) can be reused via the definition *hit_lightning* below (this definition is similar to rule (3) in Section 3.1, just adding the state information T):

$$hit\_lightning(I, O, T) \leftarrow hit\_lightning_{ab}(E, T), produce\_context(O, I, E).$$

We may observe that, like in Example 3 of Tabdual, the subgoal *storm* in the above rule of $crash_{ab}$ is called with a different actual input abductive context (viz., $E_1 = [cumulonimbus, low\_altitude]$) from that of the same subgoal in rule (6).

Finally, the different purposes of the dual program transformation, employed both in Tabdual and Evolp/r, are now consolidated within this unified approach:

- Following the name convention for dual predicates in Tabdual, the dual negation $\sim F$ of fluent literal F is now renamed to not_F in the unified approach.

- As in the positive rule transform, the abductive context and state information (viz., holds-time) jointly figure in the parameters of dual predicates. Recall Example 4. Considering *close_library* as a fluent, the dual transformation of the unified approach results in rules below (abstracted away from the two rule

name fluents of *close_library* and other implementation details):

$$\begin{aligned} not\_close\_library(C_0, C_2, Ht) \leftarrow\ & close\_library^{*1}(C_0, C_1, Ht_1), \\ & close\_library^{*2}(C_1, C_2, Ht_2), \\ & latest([Ht_1, Ht_2], Ht) \end{aligned} \tag{7}$$

$$\begin{aligned} close\_library^{*1}(I, O, Ht_{we^*}) &\leftarrow weekend^*(I, O, Ht_{we^*}). \\ close\_library^{*2}(I, O, Ht_{we}) &\leftarrow weekend(I, O, Ht_{we}). \\ close\_library^{*2}(I, O, Ht_{na}) &\leftarrow not\_absent(I, O, Ht_{na}). \end{aligned}$$

From the abduction perspective, it helps to efficiently deal with downwards by-need abduction under negated goals, e.g., abduction via $close\_library^{*i}$ due to invoking the goal $not\_close\_library([\,], O, T)$. The state when an abducible is abduced, e.g., Ht_{we^*} of $weekend^*$ in the rule of $close\_library^{*1}$, is determined in rule (7) by Ht (viz., by the reserved predicate *latest*): it can be a concrete timestamp Ht_2 or the actual query-time Qt (if Ht_2 is not concrete yet, which is the case when Ht_2 itself refers to another abduction state).

From the updating perspective, it helps to incrementally propagate upwards the dual negation complement of a fluent. For instance, updating the program with $weekend^*$ at Ht_{we^*} will incrementally propagate Ht_{we^*} to determine Ht of fluent $not\_close\_library$, in this case via $close\_library^{*1}$.

4 Counterfactuals with LP Abduction and Updating

In this section we propose an approach that makes use of LP abduction and updating for counterfactual reasoning [47]. It is based on Pearl's approach [40] to evaluate counterfactuals, and adapts it to LP. In Pearl's approach, 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 consequent's probability.

Our LP-based approach abstains from the use of probability, and concentrates on pure non-probabilistic counterfactual reasoning in LP, by resorting to LP abduction and updating, to determine the logical validity of counterfactuals under Well-Founded Semantics (WFS). Nevertheless, the approach is also adaptable to other semantics, e.g., Weak Completion Semantics [24] is employed in [44].

In the sequel, the Well-Founded Model of program P is denoted by $WFM(P)$. The logical consequence relation $\models$ is defined such that for formula F, $P \models F$ iff F is true in $WFM(P)$.

4.1 Causation and Intervention in LP

Two important ingredients in Pearl's approach of counterfactuals are causal model and intervention. Given that the inferential arrow in a LP rule representing causal direction, causation can be captured by LP abduction. It expresses a causal source by providing an explanation to a given observation. With respect to Pearl's causal model: (1) An observation O corresponds to Pearl's definition for evidence. That is, O has rules concluding it in program P, and hence does not belong to the set of abducibles A; (2) Pearl's model comprises a set of background variables (also known as exogenous variables), whose values are conditional on case-considered observed evidences and are not causally explained in the model. In terms of LP abduction, they correspond to a set of abducibles $E \subseteq A$ that provide abductive explanations to observation O. Indeed, these abducibles have no preceding causal explanatory mechanism, as they have no rules concluding them in the program.

Whereas abduction permits obtaining explanations to observations, the evaluation of counterfactual "if Pre had been true, then $Conc$ would have been true", following Pearl's approach, requires the so-called intervention. This is achieved by explicitly imposing the desired truth value of Pre, and subsequently checking whether the predicted truth value of $Conc$ 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.

LP updating complements LP abduction in establishing an apposite adjustment to the causal model by *hypothetical* updates of causal intervention on the program, affecting defeasible rules. Moreover, LP updating facilitates fixing the initial abduced background context of the counterfactual being evaluated in the program, by updating the program with the preferred explanation to the current observations.

Next, we detail the roles of LP abduction and updating in our procedure for evaluating the validity of counterfactuals.

4.2 Evaluating Counterfactuals in LP

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. The three-step procedure is detailed as follows.

Let program P encode 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:*** Perform abduction to explain past circumstances in the presence of evidence (i.e., factual observation).[2]. More to the point, compute an explanation $E \subseteq A$ to the observation O. The selected explanation is fixed (via updating) as the abduced background context in which the counterfactual is evaluated ("all other things being equal"), to obtain 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: 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*: Adjust the resulting transform $(P \cup E)_\tau$ to comply with the antecedent of the counterfactual. The antecedent of the counterfactual is imposed with an intervention on the program, by updating the program with the hypothetical fluents corresponding to the required intervention: update program $(P \cup E)_\tau$ with literal $make(B)$ or $make\_not(B)$, for $L = B$ or $L = not\ B$, resp. Assuming that Pre is consistent, $make(B)$ and $make\_not(B)$ will not 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 the consequent of the counterfactual deductively follows: $(P \cup E)_{\tau,\iota} \models Conc$. Additionally, ensure that all ICs (if any) are also satisfied.

[2] Factual observations are those explicitly given, and not presupposed from the considered counterfactual. Implicit presuppositions, either those appearing in counterfactual or indicative conditionals, cannot immediately be regarded as observations.

The counterfactual "if Pre had been true, then $Conc$ would have been true" is thus *valid* given an observation O iff O is explained by $E \subseteq A$, $(P \cup E)_{\tau,\iota} \models Conc$, and all ICs (if any) are satisfied in $WFM((P \cup E)_{\tau,\iota})$.

We illustrate in Example 8 how this three-step procedure is actually realized using a combination of LP abduction and updating to evaluate the validity of counterfactuals. The reader is referred to [47] for its formal procedure and implementation details. The procedure has been implemented in a prototype, Qualm, on top of our unified approach of LP abduction and updating (cf. Section 3.2.2), in XSB Prolog.

Example 8. Recall the example in Section 1: *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 consider more abductive causes for the forest fire: storm (which implies lightning hitting the ground) or barbecue. Note that dry leaves are important for forest fire in both cases. This example is expressed in the program below, where the set of abducibles is $A = \{storm, barbecue, storm^*, barbecue^*\}$:

$$fire \leftarrow barbecue, dry\_leaves. \qquad fire \leftarrow barbecue^*, lightning, dry\_leaves.$$
$$lightning \leftarrow storm. \qquad dry\_leaves.$$

The explicit use of $barbecue^*$ in the second rule of $fire$ 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 the factual observations is $O = \{lightning, fire\}$. Note that the observations assure us that both the antecedent and the consequent literals of the counterfactual were factually false, ensuring that the latter is relevant.

1. ***Abduction***: Abduce consistent explanations $E \subseteq A$ to the above factual observations O, viz., $E_1 = \{storm, barbecue^*\}$ and $E_2 = \{storm, barbecue\}$. Say E_1 is preferred for consideration. The abduced background context for the counterfactual is fixed by updating the program with E_1, obtaining $P \cup E_1$.

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

   $$fire \leftarrow barbecue, dry\_leaves. \qquad fire \leftarrow barbecue^*, lightning, dry\_leaves.$$
   $$lightning \leftarrow storm, not\ make\_not(lightning). \qquad dry\_leaves.$$
   $$lightning \leftarrow make(lightning).$$

   Program $(P \cup E_1)_\tau$ is updated with $make\_not(lightning)$ as the required causal intervention, viz., "if there had not been lightning".

3. ***Prediction***: Verify if the conclusion "the forest fire would not have occurred" holds in WFS. Indeed, $(P \cup E_1)_{\tau,\iota} \models not\ fire$. That is, $not\ fire$ holds with respect to the intervened modified program for explanation $E_1 =$

$\{storm, barbecue^*\}$ and the intervention $make\_not(lightning)$. Thus, the counterfactual is valid.

Example 9. Continuing Example 8, if E_2 is instead preferred to update P, the counterfactual is no longer valid. In this case, $(P \cup E_1)_\tau = (P \cup E_2)_\tau$, and the required causal intervention is also the same: $make\_not(lightning)$. But we now have $(P \cup E_2)_{\tau,\iota} \not\models not\ f$. Indeed, this conforms with our understanding that the forest fire would still have occurred but due to an alternative cause, viz., a barbecue.

Skeptical and credulous counterfactual evaluations could ergo be defined, i.e., by evaluating the proferred 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.

5 Applications to Computational Morality

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 frequent 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". See [51] for a review. Such a conception of counterfactuals in morality is discussed below by examining the issues of moral permissibility (Section 5.1) and its justification (Section 5.2).

5.1 Moral Permissibility

In this section we look into the application of counterfactuals for examining viewpoints on moral permissibility, exemplified by classic moral dilemmas from the literature on the Doctrines of Double Effect (DDE) [37] and of Triple Effect (DTE) [29]. Examining moral permissibility with counterfactuals according to these two well-known doctrines does not require formalizing them, but instead detailing, according to our approach, their materialization in concrete moral dilemmas. Classic

dilemmas regarding these doctrines are presented, so the results of counterfactual evaluation are readily comparable to those intended in the literature. Moreover, our applications focus on showing the process of moral reasoning through counterfactuals, rather than representing formally the notions of obligation, prohibition, and permissibility in a formal logic (say, as deontic logic operators). Such distinct research direction has been discussed elsewhere, e.g., [49, 7].

DDE is often invoked to explain 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 a *means* to bringing about the same good end [37]. In [22], DDE has been utilized to explain the consistency of judgments, shared by subjects from demographically diverse populations, on a series of moral dilemmas.

Counterfactuals may provide a general way to examine DDE in dilemmas, e.g., the classic trolley problem [16], 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 DDE. 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 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 others' (e.g., observers). Moreover, our emphasis on causation in this application focuses on agents' deliberate actions, rather than on causation and counterfactuals in general. See [9, 40, 23] for a more general and broad discussion on causation and counterfactuals.

Examples 10 and 11 apply this counterfactual form in two off-the-shelf military cases from [61]: terror bombing vs. tactical bombing. The former refers to bombing a civilian target during a war, thus killing civilians, in order to terrorize the enemy, and thereby get them to end the war. The latter case is attributed to bombing a military target, which will effectively end the war, but with the foreseen consequence of killing the same number of civilians nearby. According to DDE, 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 10. Terror bombing is expressed by the program below, where abducibles are $A_{teb} = \{terror\_bombing, terror\_bombing^*\}$:

$$end\_war \leftarrow terrorize\_enemy. \qquad terrorize\_enemy \leftarrow kill\_civilian.$$
$$kill\_civilian \leftarrow bomb\_civilian. \qquad bomb\_civilian \leftarrow terror\_bombing.$$

We consider *end_war* as the goal and the counterfactual "if civilians had not been killed, then the war would not have ended", provided the factual observation $O = \{kill\_civilian, end\_war\}$. It has a single explanation $E_{teb} = \{terror\_bombing\}$. Given the transform $kill\_civilian \leftarrow bomb\_civilian, not\ make\_not(kill\_civilian)$ and the intervention $make\_not(kill\_civilian)$, the counterfactual is valid, because *not end_war* holds in WFS with respect to the intervened modified program and the background context E_{teb}, i.e., $(P \cup E_{teb})_{\tau,\iota} \models not\ end\_war$. That means the morally wrong *kill_civilian* is instrumental in achieving the goal *end_war*: it is a cause for *end_war* by performing *terror_bombing* and not merely its side-effect. Hence *terror_bombing* is DDE morally impermissible.

Example 11. Tactical bombing with the same goal *end_war* is modeled by the program below, where $A_{tab} = \{tactical\_bombing, tactical\_bombing^*\}$:

$$end\_war \leftarrow bomb\_military. \qquad bomb\_military \leftarrow tactical\_bombing.$$
$$kill\_civilian \leftarrow tactical\_bombing.$$

The counterfactual is the same and now we have $E_{tab} = \{tactical\_bombing\}$ as the only explanation to the same observation $O = \{kill\_civilian, end\_war\}$. By imposing the intervention $make\_not(kill\_civilian)$ on the rule transform $kill\_civilian \leftarrow tactical\_bombing, not\ make\_not(kill\_civilian)$, one can verify that the counterfactual is not valid, because *end_war* holds in WFS with respect to the intervened modified program and the background context E_{tab}. Therefore, the morally wrong *kill_civilian* is just a side-effect in achieving the goal *end_war*. Hence *tactical_bombing* is DDE morally permissible.

This application of counterfactuals can be extended to distinguish moral permissibility according to DDE vs. DTE. DTE [29] refines DDE particularly on the notion about harming someone as an intended means. That is, DTE 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 DDE. Though DTE 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. Kamm [29] proposed DTE to accommodate a variant of the trolley problem, viz., the *Loop Case* [65]: *A trolley is headed toward five people walking on the track, and they will not be able to get off the track in time. The trolley can be redirected onto a side track, which loops back towards the five. A fat man sits on this looping side track, whose body will by itself stop the trolley. Is it*

morally permissible to divert the trolley to the looping side track, thereby hitting the man and killing him, but saving the five? This case strikes most moral philosophers that diverting the trolley is permissible [39]. Referring to a psychology study [22], 56% of its respondents judged that diverting the trolley in this case is also permissible. To this end, DTE may provide the justification, that it is permissible because it will hit the man, and not in order to intentionally hit him [29]. Nonetheless, DDE views diverting the trolley in the Loop case as impermissible.

We use counterfactuals to capture the distinct views of DDE and DTE in the Loop case.

Example 12. We model the Loop case with the program below, where *save*, *divert*, *hit*, *tst*, *mst* stand for *save the five, divert the trolley, man hit by the trolley, train on the side track* and *man on the side track*, respectively, with *save* as the goal and abducibles $A_{loop} = \{divert, divert^*\}$:

$$save \leftarrow hit. \qquad hit \leftarrow tst, mst. \qquad tst \leftarrow divert. \qquad mst.$$

DDE 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 the factual observation $O = \{hit, save\}$, its only explanation is $E_{loop} = \{divert\}$. Note that rule $hit \leftarrow tst, mst$ transforms into $hit \leftarrow tst, mst, not\ make\_not(hit)$, and the required intervention is $make\_not(hit)$. The counterfactual is therefore valid, because *not save* holds in WFS of the intervened modified program, given the background context E_{loop}. This means *hit*, as a consequence of action *divert*, is instrumental as a cause of goal *save*. Therefore, *divert* is DDE morally impermissible.

DTE 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 Example 12, we have the fact *mst* 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 (exemplified here by pushing the man – *push* for short) 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 factual observation $O = \{push, hit\}$ has no explanation $E \subseteq A_{loop}$, i.e., $push \notin A_{loop}$, 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 *hit* is a consequence of *divert* and instrumental in achieving *save*, no deliberate action is required to cause *mst*, in order for *hit* to occur. Hence *divert* is DTE morally permissible.

Example 13. Consider a variant of the Loop case, viz., the *Loop-Push Case* (cf. Extra Push Case in [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 fat man in order to place him on the side track is additionally performed. This case is modeled by the program below, where $A_{lp} = \{divert, push, divert^*, push^*\}$:

$$save \leftarrow hit. \quad hit \leftarrow tst, mst. \quad tst \leftarrow divert. \quad mst \leftarrow push.$$

Recall the counterfactuals in the discussion of DDE and DTE of the Loop case:

- "If the man had not been hit by the trolley, the five people would not have been saved." The observation $O = \{hit, save\}$ provides an extended explanation $E_{lp_1} = \{divert, push\}$, i.e., 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 $make\_not(hit)$ is applied to the same transform, resulting in a valid counterfactual, because *not save* holds in WFS of the intervened modified program with the background context E_{lp_1}.

- "If the man had not been pushed, then he would not have been hit by the trolley." The factual observation is $O = \{push, hit\}$, explained by $E_{lp_2} = \{divert, push\}$. As there is no rule for *push*, rule $push \leftarrow not\ make\_not(push)$ is introduced, and the intervention $make\_not(push)$ renders *not hit* to hold in WFS of the intervened modified program, given E_{p_2} as its background context. Thus, whereas this counterfactual is not valid in DTE of the Loop case, it is valid in the Loop-Push case.

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, DTE agrees with DDE 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.

5.2 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* [35], 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 14. Consider a scenario developed from the Loop case of the trolley problem, which happens on a particularly foggy day. Due to the 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 $divert(X)$ is an abducible atom.

$$
\begin{array}{lcll}
run\_sidetrack(X) & \leftarrow & divert(X). & \\
hit(X,Y) & \leftarrow & run\_sidetrack(X), on\_sidetrack(Y). & \\
save\_from(X) & \leftarrow & sidetrack(straight), run\_sidetrack(X). & \\
save\_from(X) & \leftarrow & sidetrack(loop), hit(X,Y), heavy\_enough(Y). & \\
sidetrack(straight) & \leftarrow & foggy. & \\
sidetrack(loop) & \leftarrow & not\ foggy. & \\
foggy. & & on\_sidetrack(man). & heavy\_enough(man).
\end{array}
$$

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 being hit by the trolley is just a side-effect of achieving the goal, and thus $divert(trolley)$ is morally permissible according to DDE. Indeed, this case resembles the original trolley problem (also commonly known as the Bystander case).

At some later time point, the fog has subsided, and by then it was clear to the agent that the sidetrack 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 his action $divert(trolley)$. For one, it can employ the aforementioned form of compound counterfactual "Had I known that the sidetrack is looping, then if I had not diverted the trolley, the five would have been saved" as a form of self-justification. Given the present knowledge that the side track is looping, the inner counterfactual is not valid, meaning that to save the five people, diverting the trolley (with the consequence of the man being hit) is required.

Moreover, the counterfactual "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 valid, meaning that this action is DDE impermissible (cf. Example 12). Therefore, the agent can justify that what would have followed (given its present knowledge, i.e., $sidetrack(loop)$) is no morally better than the actual one, when there was lack of that knowledge: its decision $divert(trolley)$ at that time was instead DDE permissible.

Qualm can evaluate such compound counterfactuals, thanks to its implemented incremental tabling of fluents (Section 3.2.1). 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, cf. [31]). 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; more recent updates are kept in tables and readily provide the current knowledge. The discussion of this technique can specifically be found in Section 3.5 of [47].

A different standpoint from where to justify its action is by resorting to Scanlon's *contractualism* [60]. Scanlon argues that moral permissibility can be addressed through the so-called *deliberative* employment of moral judgments. That is, the question of the permissibility of actions is answered by identifying the justified but defeasible argumentative considerations, and their exceptions. It is based on a view that moral dilemmas typically share the same structure: (1) They concern general principles that in some cases admit exceptions; (2) They raise questions about when those exceptions apply. With this 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. In this vein, for the example we are currently discussing, the DTE serves 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 12.

We extend now Example 14 to further illustrate how moral permissibility of actions is justified through defeasible argumentative considerations according to Scanlon's contractualism.

Example 15. As the trolley approached, the agent realized that the man was not heavy enough to stop it. This information is acknowledged by the agent through updating its knowledge base with $not\ heavy\_enough(man)$. But there was a heavy cart on the bridge over the looping sidetrack that the agent could push to place it on the side track, and thereby stop the trolley. This scenario is expressed with rules

below ($push(X)$ is an abducible atom), in addition to the program of Example 14:

$$on\_sidetrack(X) \leftarrow on\_bridge(X), push(X). \quad (8)$$
$$on\_sidetrack(Y) \leftarrow push(X), inside(Y, X). \quad (9)$$
$$on\_bridge(cart). \qquad heavy\_enough(cart).$$

The second rule of *on_sidetrack* 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 fat man, who was heavy enough, was inside the cart: the agent updates its knowledge base with $inside(fat\_man, cart)$ and $heavy\_enough(fat\_man)$. As a consequence, this man was also on the side track and hit by the trolley, which can be verified by query `?-` $hit(trolley, fat\_man)$.

In this scenario, a deliberate action of pushing was involved that consequently placed the fat man on the side track (as verified by `?-` $on\_sidetrack(fat\_man)$) 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 fat 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 DTE. In this case, the fat 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 fat man would not have been hit by the trolley", which is valid given the abduced background context $push(cart)$. Furthermore, the observation $hit(trolley, fat\_man)$ cannot be explained by $push(fat\_man)$ given the absence of the fact $on\_bridge(fat\_man)$, i.e., the hypothetical action $push(fat\_man)$ is not the causal source for the fat man being hit by the trolley.

All moral examples presented in this paper have been successfully tested in Qualm. Examples 14 and 15 together actually form one single program, where Qualm features (derived from its constituents, Tabdual and Evolp/r) are exercised. For instance, in Example 14, 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 deliberative and reactive reasoning in a form of the dual-process model [10, 33], when viewing tabling as a

form of low-level reactive behavior. Another feature used, from Evolp/r, is that rules (8) and (9) are first switched off in Example 14, via the rule name fluent mechanism (cf. Section 3.2.1), as they are not applicable in the considered scenario. Only later, in Example 15, are they switched on again to allow abducing additional action $push(cart)$.

6 Summary and Concluding Remarks

Abduction has recently been on the back burner in LP. One of our research contributions is to address challenges in engineering LP abduction systems, to revive abduction in LP. We present an implemented tabled abduction technique, Tabdual, to benefit from LP tabling mechanisms in contextual abduction, so that priorly obtained (and tabled) abductive solutions can be reused from one abductive context to another.

Aiming at facilitating the interplay between LP abduction and other LP non-monotonic reasoning, we have also combined this tabled abduction technique with our own-developed LP updating Evolp/r; the latter employs an incremental tabling feature of XSB Prolog, and 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 other contribution is that of applying our unified approach of LP abduction and LP updating to formulate counterfactuals reasoning based on Pearl's structural theory, but omitting formal probability, given the lack of pure non-probabilistic counterfactual reasoning in LP. In our LP-based counterfactual approach, abduction specifically plays an important role, to explain past circumstances in the presence of evidence (factual observation), and fix (via LP updating) considered explanations as the abduced background context in which the counterfactual is to be evaluated. Our implemented approach and its prototype Qualm is based on the Well-Founded Semantics, but adaptable to other semantics as well, e.g., Weak Completion Semantics [24] is employed in [44], where manual updating is used.

We also apply counterfactuals to model morality issues. Counterfactuals are employed to examine moral reasoning about permissibility by resorting to our LP approach, to distinguish between causes and side-effects as a result of agents' actions to achieve a goal. Counterfactuals are also used to support justifying moral permissibility of agents' actions, either by a compound counterfactual formulation or by arguing an appropriate moral principle as an applicable exception, following Scanlon's contractualism.

Side-effects in abduction have been investigated in [42, 43] through the concept

of inspection points; the latter are construed in a procedure by 'meta-abducing' a specific abducible *abduced*(A) whose function is only that of checking that its corresponding abducible A is indeed already adopted elsewhere. Therefore, the consequence of the action that triggers this 'meta-abducing' is merely a side-effect. Indeed, inspection points may be employed to distinguish a cause from a mere side-effect, and thus may provide an alternative or supplement to counterfactuals employed for the above same purpose of examining moral permissibility.

The moral examples in the present paper do not exploit the use of ICs. The use of ICs within abduction, but without counterfactuals, for morality has been explored in our related work [45]. They are particularly useful to rule out impermissible actions. One of the difficulties in using an IC to express impermissibility is that it requires the representation to be crafted in sufficient detail in order for the IC to be applicable. While we use counterfactuals to examine permissibility (so we are not bound to have a subtle problem representation), ICs can be used for other purposes, e.g., to choose among mutually exclusive abducibles. On the other hand, ICs should be treated carefully in counterfactuals, because an intervention may render ICs unsatisfiable, and hence their body's support may need to be abductively revised in order to re-impose satisfaction.

While moral examples in this paper are based on those from the literature with either their conceptual or empirical results readily available, a more systematic evaluation of counterfactual reasoning in computational morality may be considered as future work. For instance, an empirical study with human subjects, say in collaboration with cognitive scientists, may be conducted to provide insights on relevant counterfactuals in examining moral permissibility of 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 form of counterfactuals to represent and to reason about using the approach fostered in this paper.

References

[1] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, and P. Torroni. Security protocols verification in abductive logic programming. In *6th Int. Workshop on Engineering Societies in the Agents World (ESAW)*, volume 3963 of *LNCS*. Springer, 2005.

[2] J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In *JELIA 2002*, volume 2424 of *LNCS*, pages 50–61. Springer, 2002.

[3] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-3):43–70, 2000.

[4] J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming*, 4(4):383–428, 2004.

[5] J. Balsa, V. Dahl, and J. G. Pereira Lopes. Datalog grammars for abductive syntactic error diagnosis and repair. In *Proc. Natural Language Understanding and Logic Programming Workshop*, 1995.

[6] C. Baral and M. Hunsaker. Using the probabilistic logic programming language P-log for causal and counterfactual reasoning and non-naive conditioning. In *IJCAI 2007*, 2007.

[7] S. Bringsjord, K. Arkoudas, and P. Bello. Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intelligent Systems*, 21(4):38–44, 2006.

[8] R. M. J. Byrne. *The Rational Imagination: How People Create Alternatives to Reality*. MIT Press, 2007.

[9] J. Collins, N. Hall, and L. A. Paul, editors. *Causation and Counterfactuals*. MIT Press, 2004.

[10] F. Cushman, L. Young, and J. D. Greene. Multi-system moral psychology. In J. M. Doris, editor, *The Moral Psychology Handbook*. Oxford University Press, 2010.

[11] M. Denecker and D. de Schreye. SLDNFA: An abductive procedure for normal abductive programs. In *Procs. of the Joint Intl. Conf. and Symp. on Logic Programming*. The MIT Press, 1992.

[12] J. Dix. A classification theory of semantics of normal logic programs: II. weak properties. *Fundamenta Informaticae*, 3(22):257–288, 1995.

[13] T. Eiter, G. Gottlob, and N. Leone. Abduction from logic programs: semantics and complexity. *Theoretical Computer Science*, 189(1-2):129–177, 1997.

[14] K. Epstude and N. J. Roese. The functional theory of counterfactual thinking. *Personality and Social Psychology Review*, 12(2):168–192, 2008.

[15] K. Eshghi. Abductive planning with event calculus. In *Proc. Intl. Conf. on Logic Programming*. The MIT Press, 1988.

[16] P. Foot. The problem of abortion and the doctrine of double effect. *Oxford Review*, 5:5–15, 1967.

[17] T. H. Fung and R. Kowalski. The IFF procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, 1997.

[18] J. Gartner, T. Swift, A. Tien, C. V. Damásio, and L. M. Pereira. Psychiatric diagnosis from the viewpoint of computational logic. In *Procs. 1st Intl. Conf. on Computational Logic (CL 2000)*, volume 1861 of *LNAI*, pages 1362–1376. Springer, 2000.

[19] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, 1986.

[20] J. Y. Halpern and C. Hitchcock. Graded causation and defaults. British Journal for the Philosophy of Science (forthcoming), Available from `http://arxiv.org/abs/1309.1226`, 2014.

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

[22] M. Hauser, F. Cushman, L. Young, R. K. Jin, and J. Mikhail. A dissociation between moral judgments and justifications. *Mind and Language*, 22(1):1–21, 2007.

[23] C. Hoerl, T. McCormack, and S. R. Beck, editors. *Understanding Counterfactuals, Understanding Causation: Issues in Philosophy and Psychology.* Oxford University Press, 2011.

[24] S. Hölldobler and C. D. P. Kencana Ramli. Logic programs under three-valued Łukasiewicz semantics. In *ICLP 2009*, volume 5649 of *LNCS*, pages 464–478. Springer, 2009.

[25] K. Inoue and C. Sakama. A fixpoint characterization of abductive logic programs. *J. of Logic Programming*, 27(2):107–136, 1996.

[26] A. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[27] A. C. Kakas and P. Mancarella. Knowledge assimilation and abduction. In *Intl. Workshop on Truth Maintenance*, ECAI'90, 1990.

[28] A. C. Kakas and A. Michael. An abductive-based scheduler for air-crew assignment. *J. of Applied Artificial Intelligence*, 15(1-3):333–360, 2001.

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

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

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

[32] D. Lewis. *Counterfactuals.* Harvard U. P., 1973.

[33] R. Mallon and S. Nichols. Rules. In J. M. Doris, editor, *The Moral Psychology Handbook.* Oxford University Press, 2010.

[34] D. R. Mandel, D. J. Hilton, and P. Catellani, editors. *The Psychology of Counterfactual Thinking.* Routledge, New York, NY, 2005.

[35] K. D. Markman, I. Gavanski, S. J. Sherman, and M. N. McMullen. The mental simulation of better and worse possible worlds. *Journal of Experimental Social Psychology*, 29:87–109, 1993.

[36] R. McCloy and R. M. J. Byrne. Counterfactual thinking about controllable events. *Memory and Cognition*, 28:1071–1078, 2000.

[37] A. McIntyre. Doctrine of double effect. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Center for the Study of Language and Information, Stanford University, Fall 2011 edition, 2004. `http://plato.stanford.edu/archives/fall2011/entries/double-effect/`.

[38] S. Migliore, G. Curcio, F. Mancini, and S. F. Cappa. Counterfactual thinking in moral judgment: an experimental study. *Frontiers in Psychology*, 5:451, 2014.

[39] M. Otsuka. Double effect, triple effect and the trolley problem: Squaring the circle in looping cases. *Utilitas*, 20(1):92–110, 2008.

[40] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge U. P., 2009.

[41] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Counterfactual reasoning based on revising assumptions. In *ILPS 1991*, pages 566–577. MIT Press, 1991.

[42] L. M. Pereira, P. Dell'Acqua, A. M. Pinto, and G. Lopes. Inspecting and preferring abductive models. In K. Nakamatsu and L. C. Jain, editors, *The Handbook on Reasoning-Based Intelligent Systems*, pages 243–274. World Scientific Publishers, 2013.

[43] L. M. Pereira, E-A. Dietz, and S. Hölldobler. Contextual abductive reasoning with side-effects. *Theory and Practice of Logic Programming*, 14(4-5):633–648, 2014.

[44] L. M. Pereira, E.-A. Dietz, and S. Hölldobler. An abductive counterfactual reasoning approach in logic programming. Available from `http://goo.gl/bx0mIZ`, 2015.

[45] L. M. Pereira and A. Saptawijaya. Modelling Morality with Prospective Logic. In M. Anderson and S. L. Anderson, editors, *Machine Ethics*, pages 398–421. Cambridge U. P., 2011.

[46] L. M. Pereira and A. Saptawijaya. Bridging two realms of machine ethics. In J. B. White, editor, *Rethinking Machine Ethics in the Age of Ubiquitous Technology*. IGI Global, 2015.

[47] L. M. Pereira and A. Saptawijaya. Counterfactuals in logic programming with applications to agent morality. Available from `http://goo.gl/6ERgGG`, 2015.

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

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

[50] O. Ray, A. Antoniades, A. Kakas, and I. Demetriades. Abductive logic programming in the clinical management of HIV/AIDS. In *Proc. 17th. European Conference on Artificial Intelligence*. IOS Press, 2006.

[51] N. J. Roese. Counterfactual thinking. *Psychological Bulletin*, 121(1):133–148, 1997.

[52] N. J. Roese and J. M. Olson, editors. *What Might Have Been: The Social Psychology of Counterfactual Thinking*. Psychology Press, New York, NY, 2009.

[53] D. Saha. *Incremental Evaluation of Tabled Logic Programs*. PhD thesis, SUNY Stony Brook, 2006.

[54] A. Saptawijaya and L. M. Pereira. Incremental tabling for query-driven propagation of logic program updates. In *LPAR-19*, volume 8312 of *LNCS*, pages 694–709. Springer, 2013.

[55] A. Saptawijaya and L. M. Pereira. Program updating by incremental and answer subsumption tabling. In *LPNMR 2013*, volume 8148 of *LNCS*, pages 479–484. Springer, 2013.

[56] A. Saptawijaya and L. M. Pereira. Tabled abduction in logic programs (Technical Communication of ICLP 2013). *Theory and Practice of Logic Programming, Online Supplement*, 13(4-5), 2013. `http://journals.cambridge.org/downloadsup.php?file=/tlp2013008.pdf`.

[57] A. Saptawijaya and L. M. Pereira. Joint tabling of logic program abductions and updates (Technical Communication of ICLP 2014). *Theory and Practice of Logic Pro-*

gramming, Online Supplement, 14(4-5), 2014. Available from `http://arxiv.org/abs/1405.2058`.

[58] A. Saptawijaya and L. M. Pereira. The potential of logic programming as a computational tool to model morality. In R. Trappl, editor, *A Construction Manual for Robots' Ethical Systems: Requirements, Methods, Implementations*, Cognitive Technologies (forthcoming). Springer, 2015. `http://centria.di.fct.unl.pt/~lmp/publications/online-papers/ofai_book.pdf`.

[59] A. Saptawijaya and L. M. Pereira. Tabdual: a tabled abduction system for logic programs. Accepted in *IfCoLog Journal of Logics and their Applications*. Available from `http://goo.gl/lcQGes`, 2015.

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

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

[62] T. Swift. Tabling for non-monotonic programming. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):201–240, 1999.

[63] T. Swift and D. S. Warren. Tabling with answer subsumption: Implementation, applications and performance. In *JELIA 2010*, volume 6341 of *LNCS*, pages 300–312. Springer, 2010.

[64] T. Swift and D. S. Warren. XSB: Extending Prolog with tabled logic programming. *Theory and Practice of Logic Programming*, 12(1-2):157–187, 2012.

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

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

[67] J. Vennekens, M. Bruynooghe, and M. Denecker. Embracing events in causal modeling: Interventions and counterfactuals in CP-logic. In *JELIA 2010*, volume 6341 of *LNCS*, pages 313–325. Springer, 2010.

Received \jreceived