Knowledge Engineering and Reasoning with Answer Set Programs and Conditional Belief Bases

Alexander Hahn[1,*,†], Andre Thevapalan[2,†], Gabriele Kern-Isberner[1,†], Christoph Beierle[3,†], Lars-Phillip Spiegel[3,†] and Marco Wilhelm[4,†]

[1]TU Dortmund University, August-Schmidt-Straße 1, 44227 Dortmund, Germany
[2]Institute of Medical Biostatistics, Epidemiology and Informatics (IMBEI), University Medical Center of the Johannes Gutenberg University Mainz, Langenbeckstraße 1, 55131 Mainz, Germany
[3]FernUniversität in Hagen, Universitätsstraße 11, 58097 Hagen, Germany
[4]Federal Institute for Occupational Safety and Health, Friedrich-Henkel-Weg 1-25, 44149 Dortmund, Germany

Abstract

Answer set programming (ASP) and reasoning from conditional belief bases are two most prominent approaches to nonmonotonic reasoning. Both frameworks are rule-based in principle, yet substantially different with respect to syntax and semantics. Recently, approaches to combine ASP programs with conditional belief bases to prioritize answer sets have been proposed. A crucial question for applying such approaches is to decide which knowledge and beliefs should be modelled in which part of the combined knowledge base, ASP or conditionals. In this paper, we investigate the differences between encoding information in ASP vs. as conditionals in more detail. We point out general differences between the reasoning processes of ASP vs. conditionals and provide guidelines for knowledge engineering with ASP and conditionals. Moreover, we illustrate how choosing either of the two frameworks has significant effects on the resulting solutions. In particular, we focus on how factual information can be taken into account in the combined framework of ASP and conditionals in various ways. Furthermore, we build bridges between ASP and conditional reasoning by showing how conditionals with a suitable syntax can be encoded as weak constraints in ASP. Conversely, this may also help find suitable weights for weak constraints in ASP programs.

Keywords

knowledge engineering, answer set programming, ASP, priorities, conditional belief bases, ranking functions, c-representations, weak constraints

1. Introduction

Answer set programming (ASP) [1, 2] is one of the most successful applications of nonmonotonic reasoning currently. ASP is particularly well suited to solve problems with combinatorial aspects. Often, many answer sets are generated, with each one representing a possible solution. However, for some applications, only one solution needs to be selected. In order to make this decision, ranking the answer sets can be helpful. To meet this requirement, various approaches to *prioritized ASP* have been published [3, 4]. In those approaches, usually priorities are associated with program rules or atoms. Recently, approaches to generate ranks for answer sets from associated conditional belief bases have been proposed [5, 6]. The basic idea is to model plausible beliefs about "good" solutions via conditionals and then make use of an inductive conditional reasoning methodology like System Z [7] or c-representations [8] to generate a ranking function [9]. This ranking function can be used to assign ranks to answer sets which express their (im)plausibility to yield a good solution.

24th International Workshop on Nonmonotonic Reasoning, July 17–19, 2026, Lisbon, Portugal
*Corresponding author.
†These authors contributed equally.
alexander.hahn@tu-dortmund.de (A. Hahn); andre.thevapalan@uni-mainz.de (A. Thevapalan); gabriele.kern-isberner@cs.tu-dortmund.de (G. Kern-Isberner); christoph.beierle@fernuni-hagen.de (C. Beierle); lars-phillip.spiegel@fernuni-hagen.de (L. Spiegel); wilhelm.marco@baua.bund.de (M. Wilhelm)
0009-0008-6114-2594 (A. Hahn); 0000-0001-7116-9338 (A. Thevapalan); 0000-0001-8689-5391 (G. Kern-Isberner); 0000-0002-0736-8516 (C. Beierle); 0009-0001-1962-752X (L. Spiegel); 0000-0003-0266-2334 (M. Wilhelm)

Actually, ASP and conditional reasoning are two major approaches to nonmonotonic reasoning with similar modelling properties (in particular, both frameworks are perfectly suited to express rules with exceptions) yet substantially different semantics and inference techniques. So, naturally the question arises which (kind of) knowledge and beliefs[1] relevant for the problem under consideration should be modeled by which framework.

The main goal of this paper is to point out crucial differences between encoding information via ASP vs. conditionals and provide rough guidelines for when to choose which representation framework when conditionals are used to provide ranks for answer sets. More precisely, we consider combined knowledge bases $(\mathcal{P}, \Delta)$ with an ASP program $\mathcal{P}$ and a conditional belief base Δ, and ranks computed from Δ via c-representations [8] are used to prioritize answer sets of $\mathcal{P}$ following the methodology presented in [5, 6]. We discuss the different roles that $\mathcal{P}$ and Δ play in the reasoning process within a running example and by highlighting generic features of the inference processes in ASP vs. with conditionals. Due to crucial differences between the inference techniques in both frameworks and the different roles that the two components of $(\mathcal{P}, \Delta)$ play, it becomes clear that the knowledge engineering task to set up such a suitable combined knowledge base for a given problem can be challenging, and that encoding a rule via ASP or via conditionals can cause significant differences. Even factual information can be encoded in various ways and play very different roles in both frameworks, as we elaborate in more detail in this paper.

However, tackling these challenges of knowledge engineering by carefully observing what $\mathcal{P}$ and Δ are meant to express is rewarding. Our investigations show that making use of both ASP and conditionals for problem modelling offers a very rich and versatile environment for knowledge engineering that makes the quality of solutions computed by ASP explainable by conditional beliefs. The conditional belief base Δ provides a background theory for assessing the quality of solutions and hence allows for justifying why a best solution is superior to others.

Finally, we show how crucial semantic parameters of c-representations of Δ can be encoded directly in ASP via weak constraints. The basic idea here is to align the weights of weak constraints with the impact factors defining c-representations. In this way, ASP helps implementing c-representation and inferences therefrom. On the other hand, c-representations provide semantic information to help specify the weights of weak constraints in a (conditional-)logically sound way. This allows for making use of features of both frameworks in a more efficient way in ASP.

This paper is organized as follows: In the next section, we recall basic preliminaries of ASP and conditional reasoning. In Section 3, we sketch the approach on which this paper is based and explore related work. In Section 4, we discuss challenges of knowledge engineering in our combined framework and focus on propositional information in Section 5. Section 6 then shows a way to encode conditionals in ASP via weak constraints, and Section 7 concludes the paper.

2. Preliminaries

We work with a finitely generated propositional language $\mathcal{L}$ over the signature $\Sigma = \{a, b, c, \ldots\}$. The elements of Σ are called *atoms*. Formulas $A, B, C, \ldots$ over $\mathcal{L}$ are formed recursively using the standard connectives $\wedge, \vee, \neg$. For conciseness of notation, we will write AB instead of $A \wedge B$ for conjunctions, and $\overline{A}$ instead of $\neg A$ for negation. A *literal* is an atom or the negation of an atom. The symbol $\top$ denotes an arbitrary propositional tautology, and $\bot$ denotes an arbitrary contradiction.

The set of all *possible worlds* (propositional interpretations) over Σ is denoted by Ω, and $\omega \models A$ means that the propositional formula $A \in \mathcal{L}$ holds in the possible world $\omega \in \Omega$; then ω is called a *model* of A, and the set of all models of A is denoted by $\mathrm{Mod}(A)$. For propositions $A, B \in \mathcal{L}$, $A \models B$ holds iff $\mathrm{Mod}(A) \subseteq \mathrm{Mod}(B)$, as usual. Logical equivalence between formulas is denoted by $\equiv$. By slight abuse of notation, we will use ω both for the model and the corresponding conjunction of literals. Since

[1] Note that we don't make a categorical distinction between knowledge and belief in this paper. Both should be understood in the sense of "subjective knowledge". Roughly, we denote facts and strict ASP-rules as "knowledge" while usually the term "belief" is used for ASP-rules with default negation and conditionals.

$\omega \models A$ means the same for both readings of ω, no confusion will arise.

2.1. Answer Set Programming

Answer set programming (ASP, [10, 11]) is a declarative, rule-based programming language with *default negation* (also called *negation as failure* [12]). The goal of ASP is to find *answer sets* for *extended logic programs* (ELPs) in order to solve complex combinatorial problems. Formally, an ELP is a finite set of *rules* of the form $h \leftarrow a_1, \ldots, a_k, \mathsf{not}\ b_1, \ldots, \mathsf{not}\ b_m.$, where $h, a_1, \ldots, a_k, b_1, \ldots, b_m$ are literals from $\mathcal{L}$. Furthermore, h is called the *head*, and $a_1, \ldots, a_k, \mathsf{not}\ b_1, \ldots, \mathsf{not}\ b_m$ is called the *body* of the rule. A rule without a head is a *constraint*, and a rule with a head but without a body is a *fact*.

Usually, ASP works with a richer language which also allows variables. In this paper, we only work with variable-free ELPs, which are called *ground* ELPs. A ground ELP can be obtained by instantiating every rule in $\mathcal{P}$ with all possible substitutions of variables by constants; this procedure is called *grounding*. Given a (ground) ELP $\mathcal{P}$ without default negation (called *positive ELP*), a consistent set $\mathcal{S}$ of literals is an *answer set* of $\mathcal{P}$ if, for every rule "$h \leftarrow a_1, \ldots, a_k.$" in $\mathcal{P}$, it holds that $h \in \mathcal{S}$ whenever $\{a_1, \ldots, a_k\} \subseteq \mathcal{S}$, and $\mathcal{S}$ is $\subseteq$-minimal with this property. For ELPs $\mathcal{P}$ containing default negation, the answer sets are defined via the *Gelfond-Lifschitz-reduct*. Given a consistent set $\mathcal{S}$ of literals, the reduct $\mathcal{P}^{\mathcal{S}}$ is constructed as follows:

$$\mathcal{P}^{\mathcal{S}} = \{h \leftarrow a_1, \ldots, a_k. \mid h \leftarrow a_1, \ldots, a_k, \mathsf{not}\ b_1, \ldots, \mathsf{not}\ b_m. \in \mathcal{P} \text{ and } \{b_1, \ldots, b_m\} \cap \mathcal{S} = \emptyset\}.$$

A set $\mathcal{S}$ is an *answer set* of $\mathcal{P}$ if and only if $\mathcal{S}$ is an answer set of the positive ELP $\mathcal{P}^{\mathcal{S}}$. The set of all such answer sets is denoted as $\mathrm{AS}(\mathcal{P})$.

A *weak constraint* [13] is a special rule of the form

$$:\sim a_1, \ldots, a_k, \mathsf{not}\ b_1, \ldots, \mathsf{not}\ b_m. \quad [w : p] \tag{1}$$

where $a_1, \ldots, a_k, b_1, \ldots, b_m$ are literals, and w, p are nonnegative integers. The value w is called *weight*, and p is called *priority level*. One or both of w and p may be omitted, in which case the default value for both is 0.[2] In this paper, in case of $p = 0$, we denote the weak constraint (1) by "$:\sim a_1, \ldots, a_k, \mathsf{not}\ b_1, \ldots, \mathsf{not}\ b_m. \quad [w]$". The set of weak constraints in a program $\mathcal{P}$ is denoted as $\mathrm{WC}(\mathcal{P})$, and for each $r \in \mathrm{WC}(\mathcal{P})$, the weight and priority of r are denoted as $\mathrm{weight}(r)$ resp. $\mathrm{prio}(r)$.

Weak constraints are used for optimization, i.e. for finding *optimal answer sets* of a program $\mathcal{P}$, and do not constrain the set $\mathrm{AS}(\mathcal{P})$ itself. In other words, $\mathrm{AS}(\mathcal{P}) = \mathrm{AS}(\mathcal{P} \setminus \mathrm{WC}(\mathcal{P}))$. Similar to regular constraints, a weak constraint is *violated* by an answer set $\mathcal{A} \in \mathrm{AS}(\mathcal{P})$ if the body of the constraint is true with respect to $\mathcal{A}$. For each answer set $\mathcal{A} \in \mathrm{AS}(\mathcal{P})$, let

$$\mathcal{W}_{\mathcal{P}}(\mathcal{A}) = \{\!\{(\mathrm{weight}(r), \mathrm{prio}(r)) \mid r \in \mathrm{WC}(\mathcal{P}) \text{ is violated by } \mathcal{A}\}\!\}$$

denote the multiset of all weight-priority-level pairs of violated weak constraints. We can then compute for each answer set $\mathcal{A}$ a *cost* determined by the sum of weights of the weak constraints it violates at each priority level:

$$\mathrm{cost}^p_{\mathcal{P}}(\mathcal{A}) = \sum_{(w,p) \in \mathcal{W}_{\mathcal{P}}(\mathcal{A})} w\ . \tag{2}$$

If the program $\mathcal{P}$ is clear from context and $\mathrm{prio}(r) = 0$ for all weak constraints $r \in \mathcal{P}$, we adopt the shorthand $\mathrm{cost}(\mathcal{A})$. In general, the goal is to consecutively minimize the cost, i.e. the sum of weights, incurred by violating weak constraints at each priority level, starting from the highest priority and only continuing with the minimum-cost answer sets at each step. The remaining answer sets are considered *optimal* [15, 16]. More formally, let $\prec^{\mathrm{cost}}_{\mathcal{P}} \subseteq \mathrm{AS}(\mathcal{P}) \times \mathrm{AS}(\mathcal{P})$ be a relation among the answer sets of $\mathcal{P}$ defined via

$$\mathcal{A}_1 \prec^{\mathrm{cost}}_{\mathcal{P}} \mathcal{A}_2 \quad \text{iff} \quad (\exists p : \mathrm{cost}^p_{\mathcal{P}}(\mathcal{A}_1) < \mathrm{cost}^p_{\mathcal{P}}(\mathcal{A}_2) \ \text{ and } \ \forall q > p : \mathrm{cost}^q_{\mathcal{P}}(\mathcal{A}_1) = \mathrm{cost}^q_{\mathcal{P}}(\mathcal{A}_2))\ .$$

[2] This corresponds to how weak constraints are handled e.g. in clingo [14]. In DLV [13], w and p need to be positive integers. According to the ASP-Core-2 standard [15, 16], zero and negative weights are allowed as well.

Then the optimal answer sets of $\mathcal{P}$ are the minimal elements with respect to $\prec_{\mathcal{P}}^{\text{cost}}$:

$$\text{optAS}(\mathcal{P}) = \{\mathcal{A} \in \text{AS}(\mathcal{P}) \mid \nexists \mathcal{A}' \in \text{AS}(\mathcal{P}) : \mathcal{A}' \prec_{\mathcal{P}}^{\text{cost}} \mathcal{A}\} .$$

2.2. Reasoning with Conditionals

On top of the propositional language $\mathcal{L}$, we also consider *conditionals* $(B|A) \in (\mathcal{L}|\mathcal{L})$ which express statements like "If A, then *plausibly* B". The formula A is called the *antecedent*, and B is called the *consequent* of the conditional $(B|A)$. A *conditional belief base* Δ is a finite set of conditionals. A conditional establishes a plausible connection between its antecedent and consequent that is directed in the sense that the antecedent sets the context in which the consequent is evaluated. Plausibility provides a commonsense semantics that helps understanding the meaning of conditionals, in particular in contrast to material implications. Usually, plausibility means "plausible to be true", however, in more specialized environments, it can also be interpreted as, e.g., "plausible to be a good explanation" (in abductive scenarios), or "plausible to yield a good option for decision" (in inductive scenarios). In any case, in the context of the antecedent, the consequent is deemed to be more believable than its negation.

A suitable formal semantics for conditionals and conditional belief bases are provided e.g. by *Ordinal Conditional Functions* [9] (OCFs, also called *ranking functions* in general and in this paper) $\kappa : \Omega \to \mathbb{N} \cup \{\infty\}$ with $\kappa^{-1}(0) \neq \emptyset$, which assign degrees of implausibility, or surprise, to possible worlds. A ranking function κ can be lifted to formulas by $\kappa(A) := \min\{\kappa(\omega) \mid \omega \models A\}$, with $\kappa(\bot) = \min(\emptyset) = \infty$ by convention. The belief set $\text{Bel}(\kappa) = \{A \in \mathcal{L} \mid \omega \models A \text{ for all } \omega \in \kappa^{-1}(0)\}$ of κ is the set of all formulas that are satisfied by all most plausible worlds. Moreover, ranking functions can be conditionalized by propositions A from $\mathcal{L}$ via $\kappa|A(\omega) = \kappa(\omega) - \kappa(A)$, defined for all $\omega \in \text{Mod}(A)$. Conditionals are accepted by κ, written as $\kappa \models (B|A)$, in the following way:

$$\kappa \models (B|A) \quad \text{iff} \quad \kappa(A) = \infty \text{ or } \kappa(AB) < \kappa(A\overline{B}) \tag{3}$$

Thus, according to (3), a ranking function κ accepts a conditional $(B|A)$ if κ considers its antecedent A to be completely implausible, or if its verification AB is deemed to be strictly more plausible than its falsification $A\overline{B}$. A belief base Δ is accepted by κ, denoted by $\kappa \models \Delta$, if $\kappa \models (B|A)$ holds for every $(B|A) \in \Delta$; we then say that κ is a (ranking) model of Δ. A belief base Δ is *strongly consistent* (in the sense of [17]) if there is a ranking function κ with $\kappa(\omega) < \infty$ for all $\omega \in \Omega$ that is a model of Δ. A conditional belief base Δ is *weakly consistent* [18] if there is a ranking function κ accepting every conditional in Δ according to (3).

A proposition A can be accepted by a ranking function κ in two different ways. First, if A is deemed to be plausible, then $\kappa \models A$ if $\kappa(A) < \kappa(\overline{A})$, i.e., A is identified with the conditional $(A|\top)$, where $\top$ denotes any tautology. Second, if A is considered to be a certain fact, we use conditionals of the form $(\bot|\overline{A})$ to force all worlds satisfying $\overline{A}$ to be infeasible, with the intention of making A a strict belief [19]. We call the conditional $(\bot|\overline{A})$ a strict conditional belief. Note that $(\bot|\overline{A})$ is a self-contradictory conditional because every world satisfying the antecedent $\overline{A}$ falsifies $(\bot|\overline{A})$. Therefore, $(\bot|\overline{A})$ is a conditional that cannot be verified by any world. Hence, every belief base Δ with $(\bot|\overline{A}) \in \Delta$ is inconsistent in the sense of [17], because for every ranking function κ we have $\kappa(\overline{A} \wedge \bot) = \infty$, implying $\kappa(\overline{A} \wedge \bot) \not< \kappa(\overline{A} \wedge \overline{\bot})$. However, it can be weakly consistent by finding ranking models κ with $\kappa(\overline{A}) = \infty$.

For inductive reasoning [20] from a conditional belief base $\Delta = \{(B_1|A_1), \ldots, (B_n|A_n)\}$, a(n extended) *c-representation* of Δ is defined via (cf. [8])

$$\kappa_{(\Delta,\vec{\eta})}(\omega) = \sum_{\substack{1 \leq i \leq n \\ \omega \models A_i\overline{B_i}}} \eta_i \tag{4}$$

such that $\vec{\eta} = (\eta_1, \ldots, \eta_n) \in \mathbb{N} \cup \{\infty\}$ solves the following system of constraints $CR_{\Sigma}^{ex}(\Delta)$, given by the constraints $cr_i^{ex\,\Delta}$, for all $i \in \{1, \ldots, n\}$:

$$\min_{\substack{\omega \in \Omega_\Sigma \\ \omega \models A_i}} \sum_{\substack{1 \leq j \leq n \\ \omega \models A_j\overline{B_j}}} \eta_j = \infty \quad \text{or} \quad \eta_i > \min_{\substack{\omega \in \Omega_\Sigma \\ \omega \models A_iB_i}} \sum_{\substack{j \neq i \\ \omega \models A_j\overline{B_j}}} \eta_j - \min_{\substack{\omega \in \Omega_\Sigma \\ \omega \models A_i\overline{B_i}}} \sum_{\substack{j \neq i \\ \omega \models A_j\overline{B_j}}} \eta_j \qquad (cr_i^{ex\,\Delta})$$

When Δ is clear from context, we omit it and simply write $\kappa_{\vec{\eta}}$. The η_i's associated with each conditional $(B_i|A_i)$, $i \in \{1,\ldots,n\}$, are called *impact factors*, and $\vec{\eta} = (\eta_1,\ldots,\eta_n)$ is called an *impact vector*. The satisfaction of the constraint system $CR^{ex}_{\Sigma}(\Delta)$ ensures that $\kappa_{(\Delta,\vec{\eta})}$ as given by (4) is a model of Δ according to (3). Note that this extended definition of c-representations also include c-representations for weakly consistent conditional belief bases, in particular for those bases that contain strict conditional beliefs $(\bot|\overline{A})$. For strongly consistent belief bases, the range of the impact vectors is usually restricted to $\mathbb{N}$, and only the second part of the constraints $(cr_i^{ex\,\Delta})$ is used. However, for conditional belief bases that are only weakly consistent but not strongly consistent, any ranking model must assign rank ∞ to some worlds. Since this also applies to extended c-representations, in this case the first part of $(cr_i^{ex\,\Delta})$ is also essential. For further details, please see [19].

3. Related Work: Hybrid Prioritization of Answer Sets

This paper builds upon the approach first presented in [5] which combines ASP and conditional expert knowledge to prioritize answer sets. The basic idea is to use the ASP program $\mathcal{P}$ to generate answer sets specifying possible solutions, and to express plausible beliefs on what a good solution would be by a (consistent) set Δ of conditionals. Then an inductive reasoning method like System Z [7] or c-representations [8] is applied to Δ to build up a ranking model κ of Δ. $\mathcal{P}$ and Δ can (and usually will) be built over different logical languages but it is crucial that the respective signatures $\Sigma_{\mathcal{P}}$ resp. Σ_{Δ} overlap so that there is a common non-empty subsignature $\Sigma_{\cap} = \Sigma_{\mathcal{P}} \cap \Sigma_{\Delta}$ that establishes links between knowledge and beliefs expressed in both frameworks. Using these links means making the answer sets of $\mathcal{P}$ compatible with the possible worlds over which κ is defined. To this end, we make use of *marginalizations* on $\Sigma_{\cap}$ in both frameworks. For an answer set $\mathcal{A}$ and a set of (ground) atoms $\Sigma' \subseteq \Sigma$, *marginalization* on Σ' is defined as $\mathcal{A}_{|\Sigma'} = \{a \mid a \in \mathcal{A} \cap \Sigma'\} \cup \{\neg a \mid \neg a \in \mathcal{A}, a \in \Sigma'\}$. For a ranking function κ on Σ, *marginalization* on Σ' is given by $\kappa_{|\Sigma'}(\omega_{\Sigma'}) = \min_{\omega' \in \Omega:\, \omega' \models \omega_{\Sigma'}} \kappa(\omega')$, for $\omega_{\Sigma'} \in \Omega_{\Sigma'}$, where $\Omega_{\Sigma'}$ denotes the set of (partial) possible worlds defined over Σ'.

Then for each $\mathcal{A} \in \mathrm{AS}(\mathcal{P})$, the rank is computed by $\kappa_{|\Sigma_{\cap}}(\mathcal{A}_{|\Sigma_{\cap}}) := \kappa_{|\Sigma_{\cap}}(\bigwedge_{l \in \mathcal{A}_{|\Sigma_{\cap}}} l)$, from which a total preorder on $\mathrm{AS}(\mathcal{P})$ can be obtained: $\mathcal{A} \leq_{\kappa} \mathcal{A}'$ iff $\kappa_{|\Sigma_{\cap}}(\mathcal{A}_{|\Sigma_{\cap}}) \leq \kappa_{|\Sigma_{\cap}}(\mathcal{A}'_{|\Sigma_{\cap}})$. This total preorder (respectively, the ranks) expresses the (im)plausibility with which an answer set yields a good solution, where a lower rank means a (plausibly) better solution. Hence, an answer set $\mathcal{A}_i$ can be (strictly) preferred to an answer set $\mathcal{A}_j$ if $\kappa(\mathcal{A}_i) < \kappa(\mathcal{A}_j)$. The most preferred answer sets of a program $\mathcal{P}$ with respect to a ranking function κ are denoted as

$$\mathrm{AS}_{\kappa}(\mathcal{P}) := \{\mathcal{A} \in \mathrm{AS}(\mathcal{P}) \mid \nexists \mathcal{A}' \in \mathrm{AS}(\mathcal{P}) : \mathcal{A}' <_{\kappa} \mathcal{A}\}.$$

A numerical ranking of answer sets can also be obtained via weak constraints (cf. Section 2.1). Therefore, an interesting research question is in which way a correspondence between $<_{\kappa}$ and $\prec^{\mathrm{cost}}$, and hence between $\mathrm{optAS}(\mathcal{P})$ and $\mathrm{AS}_{\kappa}(\mathcal{P})$ can be established. In order to achieve this, the weak constraints in $\mathcal{P}$ need to implement the (precomputed) ranking function κ in one way or another. Such a method was proposed in [6] to prioritize answer sets according to System Z [7]. In that paper, weak constraints for the so-called tolerance partitions generated by System Z were encoded. In [21], a more profound approach to realize System Z in ASP is proposed by compiling the concept of rational closure directly into ASP rules without the need to consider ranking functions at all. In the paper at hand, we make use of c-representations [8] for inductive reasoning instead, which will allow for an elegant encoding of ranking functions by weak constraints. The advantage of encoding ranking functions directly in ASP is that the ASP solver may directly focus on finding the optimal answer sets (instead of having to find all answer sets first and computing their ranks afterwards).

Another approach to combining ASP and conditional belief bases has been proposed recently in [22]. In that paper, conditional beliefs are represented by *defeasible implications* of the form $\mathbf{T}(A) \rightarrow B$, where $\mathbf{T}(A)$ stands for *typical* instances of A. These defeasible implications are annotated with positive or negative weights, representing degrees of plausibility or implausibility, respectively. The overall approach of prioritizing answer sets via weak constraints encoding conditional beliefs is then quite

similar to the one outlined above. However, one major difference is that the weights of the defeasible implications have to be assigned manually by the knowledge engineer, whereas System Z and c-representations are able to provide suitable weights based on strong formal principles. Moreover, the focus of [22] lies more on using ASP as a tool for answering entailment queries. We, on the other hand, are interested in the answer sets themselves, and use prioritization via the conditional belief base in order to choose candidates from the set of all generated answer sets for further processing.

However, in none of those approaches, the knowledge engineering task of what (kind of) knowledge and beliefs should be modelled in which part of the combined knowledge base $(\mathcal{P}, \Delta)$ had been addressed. This is the topic of the next section.

4. Knowledge Engineering for Hybrid Modeling with ASP and Conditionals

In this section, we discuss the main question of the hybrid modeling approach presented in [5, 6] and sketched in Section 3: Which information should be encoded where, i.e., in ASP or as conditionals? Certain technical features regarding inference in both frameworks should be taken into account. Beyond that, a clear distinction between what is given and what can be plausibly expected from a good solution is crucial. This means that in a combined knowledge base $(\mathcal{P}, \Delta)$ according to [5], the focus of knowledge engineering for $\mathcal{P}$ and Δ must be clear:

- $\mathcal{P}$ describes a *given problem* and specifies technical feasibilities for possible solutions in general;
- Δ describes an *expected plausible scenario* specifying properties and dependencies among them for good solutions.

Note that both with ASP and conditionals, inference means reasoning, but since the ranks computed for the conditional belief base express the (im)plausibility of a solution, i.e., a marginalized part of an answer set, to be successful, ranks can play the role of priorities for answer sets.

This section makes use of a running example to illustrate technical features and highlight crucial differences between ASP and conditional reasoning.

Example 1 (House example). Imagine a family renovating their house. The family wants to pay attention to energy efficiency, however, their house has a historical facade which should be preserved, so they expect some peculiarities. They decide to use the combined approach of [5] using ASP and conditionals to find good, reasonable renovation decisions. But they feel uncertain about which framework to use for which parts of their knowledge and beliefs.

We now take a more formal look on this scenario. The family would like to make their house as energy-efficient (e) as possible. Two concrete options the family considers for the house renovation are the installation of thermal insulation (i) and triple-glazed windows (t). Moreover, they have learned about the following rule-based information:

(A) The installation of both thermal insulation and triple-glazed windows usually ensures that the house complies with energy-efficiency standards [T]. The other way round, usually energy-efficient houses are expected to be insulated and have triple-glazed windows [E].

(B) The installation of thermal insulation without triple-glazed windows is not efficient because the windows will leak the heat. Therefore, thermal insulation usually comes along with triple-glazed windows [E].

(C) The installation of triple-glazed windows without thermal insulation may lead to mold formation. This would cause serious health problems (h) [T]. Therefore, if no thermal insulation is possible then triple-glazed windows are strongly not recommended [E]. Moreover, as a strict constraint, no renovation measure should lead to serious health problems [T].

(D) If houses have historical facades (f), then thermal insulation is usually not installed [E].

	ω	impacts	κ_1	κ_2	κ_3		ω	impacts	κ_1	κ_2	κ_3
$(\mathcal{A}_1)$	$eitf$	η_6	2	2	2		$\overline{e}itf$	$\eta_1+\eta_6+\eta_7$	3	3	3
	$eit\overline{f}$	0	0	0	0		$\overline{e}it\overline{f}$	$\eta_1+\eta_7$	1	1	1
$(\mathcal{A}_2)$	$ei\overline{t}f$	$\eta_3+\eta_4+\eta_6$	3	3	3	$(\mathcal{A}_3)$	$\overline{e}i\overline{t}f$	$\eta_1+\eta_4+\eta_6$	3	4	3
	$ei\overline{t}\,\overline{f}$	$\eta_3+\eta_4$	1	1	1		$\overline{e}i\overline{t}\,\overline{f}$	$\eta_1+\eta_4$	1	2	1
	$e\overline{i}tf$	$\eta_2+\eta_5$	2	2	2		$\overline{e}\overline{i}tf$	$\eta_1+\eta_5$	3	2	2
	$e\overline{i}t\overline{f}$	$\eta_2+\eta_5$	2	2	2		$\overline{e}\overline{i}t\overline{f}$	$\eta_1+\eta_5$	3	2	2
$(\mathcal{A}_5)$	$e\overline{i}\,\overline{t}f$	$\eta_2+\eta_3$	1	1	2	$(\mathcal{A}_4)$	$\overline{e}\overline{i}\,\overline{t}f$	η_1	1	1	1
	$e\overline{i}\,\overline{t}\,\overline{f}$	$\eta_2+\eta_3$	1	1	2		$\overline{e}\overline{i}\,\overline{t}\,\overline{f}$	η_1	1	1	1

Table 1
Ranking functions κ_1, κ_2, and κ_3 for Example 1.

In this description of the renovation problem, we can roughly distinguish two kinds of information: The first kind is factual information, i.e., information concerning the concrete problem instance, like that the family owns a house with a historical facade, and information about technical feasibility, e.g. the information about compliance with energy-efficiency standards and the problem with absent insulation and triple-glazed windows. Also the information that any renovation measure causing health problems must be avoided restricts technical feasibility. This kind of information is marked with [T] for "technical domain knowledge".

The second kind of information consists of soft constraints expressing recommendations, e.g. no triple-glazed windows when thermal insulation is absent, or capturing expectations about what may yield a good solution, like trying to have both thermal insulation and triple-glazed windows to make the house as energy-efficient as possible. In the list above, this information is marked with [E] for "expectation or recommendation".

The family sets up the following ELP $\mathcal{P}$ with $\Sigma_\mathcal{P} = \{e, i, t, c_1, h, f\}$ and conditional belief base Δ with $\Sigma_\Delta = \{e, i, t, f\}$ to model their renovation scenario:

$$\begin{aligned}\mathcal{P} = \{\quad & r_1 : e \leftarrow \mathsf{not}\ \overline{e}., \quad r_2 : \overline{e} \leftarrow \mathsf{not}\ e., \quad r_3 : i \leftarrow \mathsf{not}\ \overline{i}., \quad r_4 : \overline{i} \leftarrow \mathsf{not}\ i., \quad r_5 : t \leftarrow \mathsf{not}\ \overline{t}.,\\ & r_6 : \overline{t} \leftarrow \mathsf{not}\ t., \quad r_7 : e \leftarrow i, t, \mathsf{not}\ c_1., \quad r_8 : h \leftarrow \overline{i}, t., \quad r_9 : \ \leftarrow h., \quad r_{10} : f. \qquad \}.\end{aligned}$$

The rules r_1 – r_6 just generate all possible combinations of e, i, t to be potential parts of the answer sets. Rule r_7 encodes the first part of information (A) where, c_1 represents exceptional cases like a broken roof. Rule r_8 formalizes the first part of (C), and the constraint r_9 will help ensuring the last part of (C), i.e., avoiding health problems. Fact r_{10} represents the given fact that the house has a historical facade. Furthermore, the family models the soft constraints as a conditional belief base

$$\Delta = \{\delta_1, \ldots, \delta_6\} \ \text{ with } \ \delta_1 = (e|\top),\ \delta_2 = (i|e),\ \delta_3 = (t|e),\ \delta_4 = (t|i),\ \delta_5 = (\overline{t}|\overline{i}),\ \delta_6 = (\overline{i}|f).$$

The conditional δ_1 represents the general belief that having an energy-efficient house would be a good solution. Conditionals δ_2 and δ_3 relate energy efficiency to insulation and triple-glazed windows and hence encode the second part of (A). Conditional δ_4 represents (B), and δ_5 encodes part of (C). Finally, δ_6 expresses (D).

First, we compute the answer sets of $\mathcal{P}$. The program has the following five answer sets:

$$\mathcal{A}_1 = \{e, i, t, f\},\ \mathcal{A}_2 = \{e, i, \overline{t}, f\},\ \mathcal{A}_3 = \{\overline{e}, i, \overline{t}, f\},\ \mathcal{A}_4 = \{\overline{e}, \overline{i}, \overline{t}, f\},\ \mathcal{A}_5 = \{e, \overline{i}, \overline{t}, f\}.$$

In order to rank these answer sets according to the conditional beliefs in Δ, c-representations are used to set up suitable ranking functions according to (4) and $CR^{ex}_\Sigma(\Delta)$. There are three pareto-minimal impact vectors solving $CR^{ex}_\Sigma(\Delta)$: $\vec{\eta}_1 = (1,0,1,0,2,2), \vec{\eta}_2 = (1,1,0,1,1,2), \vec{\eta}_3 = (1,1,1,0,1,2)$. The ranking functions κ_1, κ_2, and κ_3 which are induced by $\vec{\eta}_1$, $\vec{\eta}_2$, and $\vec{\eta}_3$, respectively, are shown in Table 1. From this, we obtain the ranks of the answer sets of $\mathcal{P}$ as indicated in Table 1. According to κ_1 and κ_2, the answer sets $\mathcal{A}_4$ and $\mathcal{A}_5$ are the best solutions to the renovation problem, while κ_3 considers

only $\mathcal{A}_4$ to be the most plausible solution. Note that all answer sets make use only of atoms from the common signature Σ_Δ. Hence we obtain

$$\mathrm{AS}_{\kappa_1}(\Delta) = \mathrm{AS}_{\kappa_2}(\Delta) = \{\mathcal{A}_4, \mathcal{A}_5\}, \qquad \mathrm{AS}_{\kappa_3}(\Delta) = \{\mathcal{A}_4\}.$$

According to all three ranking models, the preferred answer sets recommend to not have thermal insulation nor triple-glazed windows. Whether this meets energy-efficiency standards is unclear but not excluded. Maybe other measures, like e.g. interior thermal insulation, can help meet these standards. However, from a skeptical point of view, the presence of κ_3 suggests that the renovated house will not be energy-efficient.

Most decisively, $\mathcal{A}_4$ and $\mathcal{A}_5$ are the only answer sets that do not violate δ_6 which turned out to be the most crucial conditional, expressing the conflict between the general goal to have an energy-efficient house and the given fact that the house has a historical facade.

In Example 1, it is apparent that $\mathcal{P}$ and Δ play different roles. We summarize our observations here:

- The information in $\mathcal{P}$ is used to determine the search space of *technically feasible solutions*, given the specific problem instance, e.g., "$f.$". $\mathcal{P}$ is also used to rule out unwanted options via strict constraints (e.g., $\leftarrow h.$).
- The information in Δ describes the plausible properties of good resp. the *best solutions*. Hence the ranks computed from Δ can be understood as degrees of (in)appropriateness of answer sets to provide best solutions. Therefore, minimal ranks indicate best possible solutions.
- The conditionals from Δ can also help *explaining* why answer sets provide good or not so good solutions. In the house example, the conflict between e and f might have been anticipated, but with unclear impact. This is elaborated in a formal way by c-representations which clearly indicate that complying with δ_6 is most crucial for a low rank.

Moreover, there are general and crucial technical differences between ASP and conditionals regarding their inference behavior. Without facts or default assumptions, nothing can be entailed from an ASP program. That is why we used r_1 – r_6 in $\mathcal{P}$ to generate these possible options (or their violation) within answer sets. To the contrary, propositional beliefs can be always inferred from (consistent) conditional belief bases. The belief set of a ranking function is the theory of the worlds with rank 0. This means that from Table 1 and hence from Δ, the propositional belief $eit\overline{f}$ can be inferred. While inference in ASP is rule-based in principle (i.e., whenever the body of a rule is satisfied by an answer set, also its head must be taken to be true), inference from a set of conditionals via c-representations is more like an interactive process aiming at finding an equilibrium among the impacts of conditionals. This is expressed by $CR^{ex}_{\Sigma}(\Delta)$.

Given these crucial differences, moving information from $\mathcal{P}$ to Δ, or vice versa, can drastically change the results in a non-predictable way. Therefore, general formal results about the effects of variations of the modelling of information via ASP vs. conditionals are hardly expectable. Moreover, this is more than a technical question. In the first place, the appropriateness of the knowledge engineering for the problem under consideration should be considered. Here we hope that our insights summarized above, in particular the distinction between knowledge about technical feasibility and plausible (conditional) beliefs about "good" solutions, can be helpful.

To illustrate, on the one hand, the problems when information is moved from the ASP program to the conditional belief base, or vice versa, but also on the other hand, the richness of modelling options that is offered by this combined framework of ASP and conditionals, we focus on the modelling of factual resp. propositional information in the next section.

5. Propositional Information in ASP and Conditional Belief Bases

Both ASP programs and conditional belief bases are excellent formal means to encode generic knowledge in the form of default rules, i.e., rules that express important associations but also allow for exceptions.

Such rules are indispensable to represent commonsense and domain-specific knowledge. Nevertheless, also facts and other propositional information that is syntactically simpler than (default) rules are deemed essential to model knowledge, e.g., to take observations or given prerequisites into account. They may complement the default rules in various ways. However, as we argued in Section 4, ASP rules and conditionals follow substantially different reasoning paradigms. Therefore, encoding knowledge and beliefs via ASP or conditionals can cause significant differences. The research question that we focus on in this section is whether this also applies to propositional information, i.e., whether encoding propositional information in the ASP program $\mathcal{P}$ or associating it with the conditional belief base Δ matters. We will see that also in this apparently simple case, many options to integrate propositional information into the combined knowledge base $(\mathcal{P}, \Delta)$ exist, and crucial differences between ASP encoding vs. conditional encoding may occur. First, we give an overview on these modelling options. Afterwards, we contrast some of these options to highlight striking differences.

5.1. A Variety of Modelling Options

In this section, we assume a combined knowledge base $(\mathcal{P}, \Delta)$ consisting of an ASP program $\mathcal{P}$ and a conditional belief base Δ to be given. The knowledge engineering task that we consider is to take some additional propositional information $A \in \mathcal{L}$ into account.

The proposition A can express a certain fact, or a plausible belief; it can point out something that must be avoided, like in ASP constraints, or represent case data to which generic knowledge and beliefs should be applied. So, the meta information that comes along with A plays a crucial role for deciding how to integrate A in the knowledge engineering process. We discuss the various options for representing A suitably in the following, where the different focuses of modelling of $\mathcal{P}$ resp. Δ should be taken into account.

1. If A is a certain fact, then adding "A." (if A is a literal) to the ASP program $\mathcal{P}$ is a suitable option. But also in the conditional belief base Δ, a certain fact A can be expressed by the strict conditional belief $(\bot|\neg A)$ that ensures that only models of A can have a finite rank. The fact "A." in $\mathcal{P}$ represents a given fact of the problem description, while the conditional $(\bot|\neg A)$ in Δ marks models of $\neg A$ to be infeasible solutions.
2. Plausible beliefs A can be represented by conditionals $(A|\top)$. As elements of Δ, they ensure that the minimal worlds (according to any ranking model) all satisfy A. Such conditionals can be used to express general properties of good solutions, like the conditional $(e|\top)$ in Example 1.
3. As a meta-level constraint for suitable solutions that must be avoided, if A is a literal, it can be encoded by the (technical) constraint "$\leftarrow\ A$" in ASP. In this way, the constraint restricts technically feasible solutions already in the generation process.
4. If A represents case data to which the generic knowledge and belief in $\mathcal{P}$ or Δ should be applied, then the c-representation κ which is generated from Δ can be conditionalized by A. As an analogue procedure on the ASP side, only answer sets of $\mathcal{P}$ which contain A (if it is a literal) can be considered; this can be encoded by adding the constraint "$\leftarrow$ not A." to $\mathcal{P}$.

This enumeration shows that propositional information can influence the modelling process in very different ways. In the rest of this section, we elaborate on the technical implications of these modelling options in ASP or via conditionals, respectively, for knowledge engineering.

5.2. ASP Facts vs. Strict and Plausible Conditional Beliefs

Literal facts A in ASP are established via fact rules "A.", enforcing every answer set to contain the literal A. Adding a fact to an ASP program $\mathcal{P}$ may block some rules that were previously used or may lead to new rules being activated that previously were not used for generating answer sets. This may lead to completely different answer sets, fully in line with the nonmonotonic characteristics of ASP. In conditional belief bases, the literal fact $A = f$ can be enforced via the conditional $(\bot|\overline{f})$, making all worlds satisfying $\overline{f}$ infeasible and thus enforcing the belief f. Adding $(\bot|\overline{f})$ to a belief base Δ may

influence the impacts assigned to the other conditionals of Δ. We give an example on how the modeling of f influences both semantics and the resulting prioritization of answer sets.

Example 2. Consider the following answer set program $\mathcal{F} = \{ \quad \overline{a} \quad \leftarrow f, \mathsf{not}\ a. \quad a \quad \leftarrow \mathsf{not}\ \overline{a}.\}$ with answer set $A_1 = \{a\}$ and $\Delta_f = \{(\overline{f}|a)\}$ with pareto minimal impact vector $\vec{\eta} = (1)$, yielding the c-representation κ (cf. Table 2). Modeling f in Δ_f yields $\Delta'_f = \{(\overline{f}|a), (\bot|\overline{f})\}$ with pareto minimal impact vector $\vec{\eta} = (\infty, \infty)$, yielding the (extended) c-representation κ' (cf. Table 2). Then $\kappa'(A_1) = 0$ and thus A_1 is a preferred answer set of $\mathcal{F}$ with respect to κ', i.e., a occurs in all prioritized answer sets of $\mathcal{F}$ with respect to κ'. Modeling the fact f in ASP instead yields the program $\mathcal{F}' = \mathcal{F} \cup \{f.\}$ which has the answer sets $A'_1 = \{a, f\}$ and $A'_2 = \{\overline{a}, f\}$ with $\kappa(A'_2) = 0 < 1 = \kappa(A'_1)$, thus A'_2 is a preferred answer set of $\mathcal{F}'$ with respect to κ. Not only does a not occur in all prioritized answer sets of $\mathcal{F}'$ with respect to κ, but $\overline{a}$ occurs instead.

Example 2 shows that the method chosen to model f drastically influences the resulting answer sets. It is therefore not possible to simply switch from modeling facts in ASP to modeling facts as strict conditional beliefs and vice versa.

Since an ASP fact "A." forces A to appear in all answer sets, adding $(A|\top)$ to Δ may seem less problematic than adding a strict conditional belief $(\bot|\overline{A})$. However, basically the same issues apply: The additional conditional in Δ may influence the impacts assigned to the other conditionals. Moreover, even plausible conditional beliefs can lead to inconsistencies. For example, adding $(f|\top)$ to the conditional belief base Δ in House Example 1 will make Δ (both strongly and weakly) inconsistent.

5.3. Plausible vs. Strict Conditional Beliefs and Conditionalization

In the context of conditional belief bases, A can be any classical logical formula, not only a literal.

Adding the plausible conditional belief $(A|\top)$ to Δ enforces that A must be more plausible than $\overline{A}$, i.e., most plausible. This however does not make $\overline{A}$ totally infeasible, in contrast to adding the strict conditional belief $(\bot|\overline{A})$. This latter conditional ensures that all models of $\overline{A}$ will be assigned rank ∞, hence making A a strict belief. If the plausible conditional belief $(A|\top)$ is added and $\Delta \cup \{(A|\top)\}$ is consistent, it may very well be the case that there are models κ of this extended conditional belief base with $\kappa(\omega) < \infty$ for some ω with $\omega \models \overline{A}$ and thus $\kappa(\overline{A}) < \infty$.

The conditional belief bases $\Delta \cup \{(A|\top)\}$ and $\Delta \cup \{(\bot|\overline{A})\}$ usually generate completely different ranking functions. For instance, adding $(i|\top)$ to Δ in Example 1 would not change the pareto-minimal c-representations $\kappa_1, \kappa_2, \kappa_3$ (see Table 1). The same observation holds when adding $(t|\top)$ or $(\overline{f}|\top)$ to Δ. On the other hand, adding any of the three counter-conditionals to Δ, i.e., any of $(\overline{i}|\top)$, $(\overline{t}|\top)$ or $(f|\top)$, yields a conditional belief base Δ' that is inconsistent without any ranking model.

Regarding strict conditional beliefs $(\bot|\overline{A})$, note that adding any such conditional to Δ makes the extended conditional belief base $\Delta \cup \{(\bot|\overline{A})\}$ only weakly consistent. Strong consistency is necessarily lost (see Section 2.2). But sometimes, even weak consistency can be destroyed. In Example 1, adding any of the three conditionals $(\bot|\overline{i})$, $(\bot|\overline{t})$ or $(\bot|f)$ to Δ yields a belief base Δ' that is not (strongly) consistent, but still weakly consistent. On the other hand, adding any of these three conditionals with negated antecedent, i.e., any of $(\bot|i)$, $(\bot|t)$ or $(\bot|\overline{f})$ yields a belief base Δ' that is neither strongly consistent nor weakly consistent.

This also illustrates that conditionalization of the c-representation of a conditional belief base is very different from adding a strict conditional belief to it. While in the House Example 1, $\Delta \cup \{(\bot|i)\}$ is inconsistent and hence does not allow for computing an extended c-representation at all, the conditionalized c-representations $\kappa_1|\overline{i}, \kappa_2|\overline{i}, \kappa_3|\overline{i}$ can be computed straightforwardly from Table 1. Furthermore, it is obvious that no plausible belief can yield the effect of conditionalization because adding $(A|\top)$ to Δ makes $\overline{A}$ worlds only less plausible (if the extended conditional belief base is consistent) while conditionalization excludes all models of $\overline{A}$.

5.4. ASP Constraints vs. Strict Conditional Beliefs

In this subsection, we assume $A = h$ to be a literal in compliance with the ASP syntax rules.

ω	impacts	κ	κ'	ω	impacts	κ	κ'
af	η_1	1	∞	$a\overline{f}$	η_2	0	∞
$\overline{a}f$	0	0	0	$\overline{a}\overline{f}$	η_2	0	∞

Table 2
Ranking functions κ, κ' for Example 2.

ω	impacts	κ	κ'	ω	impacts	κ	κ'
ah	η_2	0	∞	$a\overline{h}$	η_1	1	∞
$\overline{a}h$	η_2	0	∞	$\overline{a}\overline{h}$	0	0	0

Table 3
Ranking functions κ, κ' for Example 3.

While the constraint "$\leftarrow h$." and the strict conditional belief $(\bot|h)$ have similar motivation in that they are meant to make the literal h infeasible, there are still some crucial differences to consider. For a ranking function κ and a formula A, we always have $\min\{\kappa(A), \kappa(\overline{A})\} = 0$ due to the condition $\kappa^{-1}(0) \neq \emptyset$. Therefore, making h infeasible, i.e., enforcing $\kappa(h) = \infty$ by means of $(\bot|h)$, requires $\kappa(\overline{h}) = 0$ and thus also enforces that $\overline{h}$ is (strictly) believed. This is not the case in ASP, where "$\leftarrow h$." has no direct influence on whether $\overline{h}$ occurs in an answer set or not. Furthermore, while "$\leftarrow h$." acts as a filter on answer sets, it does not interact directly with the other rules of an answer set program. The conditional $(\bot|h)$ on the other hand may interact with other conditionals in Δ, drastically affecting the ranks of all worlds, not only those where h holds. We give an example illustrating this.

Example 3. Consider the following answer set program

$$\mathcal{H} = \{a \leftarrow \mathsf{not}\ \overline{a}, \mathsf{not}\ h., \quad \overline{a} \leftarrow \mathsf{not}\ a, \mathsf{not}\ h., \quad h \leftarrow \mathsf{not}\ \overline{a}, \mathsf{not}\ a., \quad \leftarrow h.\}$$

and the belief base $\Delta_h = \{(h|a)\}$. Here h is made impossible in the answer sets via the constraint "$\leftarrow h$." in $\mathcal{H}$. The program $\mathcal{H}$ has two answer sets, $\mathcal{A}_1 = \{a\}$ and $\mathcal{A}_2 = \{\overline{a}\}$, and $CR^{ex}_{\Sigma}(\Delta_h)$ has one minimal solution, $\vec{\eta} = (1)$, with induced ranking function κ (cf. Table 3). Then $\kappa(\mathcal{A}_1) = \kappa(\mathcal{A}_2) = 0$, i.e., both answer sets are preferred. We compare this to modeling the infeasibility of h in Δ_h instead of $\mathcal{H}$, yielding the program $\mathcal{H}' = \mathcal{H}\backslash\{\leftarrow h.\}$ and the belief base $\Delta'_h = \{(h|a), (\bot|h)\}$. The program $\mathcal{H}'$ has answer sets $\mathcal{A}_1 = \{a\}, \mathcal{A}_2 = \{\overline{a}\}$, and $\mathcal{A}_3 = \{h\}$, and $CR^{ex}_{\Sigma}(\Delta'_h)$ has again only one minimal solution $\vec{\eta}' = (\infty, \infty)$ with induced ranking function κ' (cf. Table 3). Then $\kappa'(\mathcal{A}_2) = 0$, and $\kappa'(\mathcal{A}_1) = \kappa'(\mathcal{A}_3) = \infty$, i.e., $\mathcal{A}_1$ is no longer preferred.

Example 3 shows how the conditional $(\bot|h)$ in Δ'_h may affect the ranking of other worlds by enforcing the impact ∞ for not only itself, but also the conditional $(h|a)$, yielding a direct effect on the prioritization of answer sets.

5.5. ASP Facts and Constraints vs. Rank Conditionalization

Another way to take facts into account in the context of conditional belief bases is the conditionalization of the induced ranking function with a formula A. If $A = f$ is a literal, this is similar to considering only those answer sets where A holds, but does not limit the computation of answer sets in any way. It can be shown that the occurrence of a literal f in all answer sets, e.g., due to a fact "f.", implies that the conditionalization of κ by f is sufficient to obtain the priority relation among answer sets.

Proposition 1. *If $f \in \bigcap \mathrm{AS}(\mathcal{P})$, then $\mathcal{A}_i \preceq_\kappa \mathcal{A}_j$ iff $\mathcal{A}_i \preceq_{\kappa|f} \mathcal{A}_j$ for all $\mathcal{A}_i, \mathcal{A}_j \in \mathrm{AS}(\mathcal{P})$.*

Proof. Let $f \in \bigcap \mathrm{AS}(\mathcal{P})$. Then we have $\mathcal{A}_i \models f$ and $\mathcal{A}_j \models f$, and thus both $\kappa(\mathcal{A}_i) \geq \kappa(f)$ and $\kappa(\mathcal{A}_j) \geq \kappa(f)$. Therefore, $\kappa(\mathcal{A}_i) \leq \kappa(\mathcal{A}_j)$ holds if and only if $\kappa(\mathcal{A}_i) - \kappa(f) \leq \kappa(\mathcal{A}_j) - \kappa(f)$, which by definition of conditionalization is equivalent to $\kappa|f(\mathcal{A}_i) \leq \kappa|f(\mathcal{A}_j)$. Altogether, we therefore have $\mathcal{A}_i \preceq_\kappa \mathcal{A}_j$ iff $\mathcal{A}_i \preceq_{\kappa|f} \mathcal{A}_j$. □

With respect to constraints, first note that we cannot model the constraint "$\leftarrow h$" via conditionalization with $\overline{h}$, as the absence of h in an answer set does not imply the occurrence of $\overline{h}$ in it.

However, as mentioned in the beginning of this section, the constraint "$\leftarrow$ not A." may be considered as an alternative for modeling conditionalization by a literal $A = \overline{h}$. This constraint eliminates all answer sets which do not contain A, which results in Proposition 1 being applicable. Similar to

conditionalization, constraints of this shape lead to a focusing on (existing) answer sets which contain A without interfering with the other rules. This is what distinguishes them from facts "A.", which directly lead to the insertion of A into answer sets and may enable the activation or blocking of rules.

5.6. Summary and Discussion

Similarly to what we discussed in Section 4, even in the case of propositional information, there is no straightforward way to move information suitably from the ASP program $\mathcal{P}$ to the conditional belief base Δ, or vice versa in general. The roles that $\mathcal{P}$ and Δ play in the reasoning and ranking process are very different, and also the reasoning modes show significant differences. While propositions as facts are an essential part of ASP reasoning, they are neither necessary nor unproblematic for conditional belief bases since they can easily lead to (partial) inconsistencies. Moreover, we observed fundamental differences between encoding a proposition A as a plausible belief via the defeasible embedding $(A|\top)$, or as a strict conditional belief via the strict embedding $(\bot|\overline{A})$. On the other hand, conditionalizing a ranking function κ generated from Δ by a proposition A representing case data is easily possible, and maybe the most frequent case where propositional information becomes relevant. Proposition 1 showed that conditionalization of κ by (e.g.) a fact A in $\mathcal{P}$ does not change the ranking of answer sets.

Constraints play a particularly interesting role for knowledge engineering with a combined knowledge base $(\mathcal{P}, \Delta)$. Technically, they must be part of $\mathcal{P}$ because there is no way to represent them adequately with conditionals, as Example 3 illustrates. And indeed, they restrict the technically feasible solutions. However, from a knowledge engineering perspective, they are more like goals which express what to avoid in a suitable solution and hence part of an expected solution. This illustrates that knowledge engineering sometimes should extend the focus of modelling for each component $\mathcal{P}$ or Δ to benefit from technical options of the other component.

In the next section, we reconsider the connection between ASP constraints and conditionals, presenting weak constraints as a suitable means to encode conditionals via ASP.

6. From Conditional Knowledge Bases to ASP with Weak Constraints

In this section, we will show that in some scenarios, a conditional knowledge base can be encoded as weak constraints directly in an ASP program. As a first step, we define the following (partial) ELP:

$$\mathcal{P}_\Sigma = \{a \leftarrow \mathsf{not}\ \overline{a}.\mid a \in \Sigma\} \cup \{\overline{a} \leftarrow \mathsf{not}\ a.\mid a \in \Sigma\}.$$

If $\mathcal{P}_\Sigma$ is a part of a larger program $\mathcal{P}$, the sub-program $\mathcal{P}_\Sigma$ ensures that every valid combination of literals from Σ is represented among the potential answer sets of $\mathcal{P}$. Then all answer sets of $\mathcal{P}$ will correspond directly to possible worlds from Ω. The next theorem demonstrates that weak constraints can be employed to encode conditionals in Δ, thereby ranking the answer sets of an ELP $\mathcal{P}$ according to a c-representation of Δ.

Theorem 2. *Let $\mathcal{P}$ be an arbitrary ELP without weak constraints such that $\mathcal{P}_\Sigma \subseteq \mathcal{P}$, and let the knowledge base Δ be given as follows:*

$$\Delta = \{(l_1^i \vee \ldots \vee l_{q_i}^i | k_1^i \wedge \ldots \wedge k_{p_i}^i) \mid 1 \leq i \leq n\}, \tag{5}$$

where all k_j^i and l_j^i are from $\mathcal{L}$. Let the ELP $\mathcal{P}_\Delta$ be defined as follows:

$$\mathcal{P}_\Delta = \{:\sim k_1^i, \ldots, k_{p_i}^i, \mathsf{not}\ l_1^i, \ldots, \mathsf{not}\ l_{q_i}^i.\quad [w_i] \mid 1 \leq i \leq n\}.$$

If $\vec{w} = (w_1, \ldots, w_n)$ is chosen such that $\kappa_{\vec{w}}$ is a c-representation of Δ, we have $\mathrm{optAS}(\mathcal{P} \cup \mathcal{P}_\Delta) = \mathrm{AS}_{\kappa_{\vec{w}}}(\mathcal{P})$.

Proof. Let $S \in \mathrm{AS}(\mathcal{P} \cup \mathcal{P}_\Delta)$. A weak constraint ":$\sim k_1, \ldots, k_p, \mathsf{not}\ l_1, \ldots, \mathsf{not}\ l_q.\ [w]$" in Δ is violated by S if S both contains $k_1, \ldots, k_p$ and does not contain any of $l_1, \ldots, l_q$. Because of the rules in $\mathcal{P}_\Sigma$,

this means $S \models k_1 \wedge \cdots \wedge k_p \wedge \bar{l}_1 \wedge \cdots \wedge \bar{l}_q$. In other words, if S violates the weak constraint above, then S implies the falsification of the conditional $(l_1 \vee \ldots \vee l_q | k_1 \wedge \ldots \wedge k_p)$. Then, from Equations (2) and (4), $\mathrm{cost}(S) = \kappa_{\vec{w}}(S)$ follows immediately. Thus, for any pair of answer sets S and S', we obtain $S \prec^{cost} S'$ iff $S <_\kappa S'$. Consequently, $\mathrm{optAS}(\mathcal{P} \cup \mathcal{P}_\Delta) = \mathrm{AS}_{\kappa_{\vec{w}}}(\mathcal{P})$ holds. □

Note that the specific form of the conditionals in (5) is not mandatory when translating conditional knowledge bases into weak constraints in the sense of Theorem 2. However, for general conditionals $(B_i|A_i)$ with A_i and B_i being arbitrary propositions, the translation process is a bit more complex due to the syntactical restriction of ASP rules which employ literals only. The basic idea is to introduce fresh atoms for A_i and B_i, say $F(A_i)$ and $F(B_i)$, and translate $(B_i|A_i)$ into the weak constraint "$:\sim F(A_i), \mathsf{not}\ F(B_i).\ [\eta_i]$", denoted by wc_i, where η_i is the impact factor of $(B_i|A_i)$ in the c-representation under consideration. In addition, A_i and B_i are compiled into any equivalent disjunctive normal form, say $\mathsf{DNF}(A_i)$ and $\mathsf{DNF}(B_i)$, respectively. For each disjunct $a_1 \wedge \ldots \wedge a_m$ in $\mathsf{DNF}(A_i)$, the ASP rule "$F(A_i) \leftarrow a_1, \ldots, a_m.$" is added to the ASP program, as well. The same has to be done for $\mathsf{DNF}(B_i)$. This guarantees that whenever A_i is true at least one of the rules of the form "$F(A_i) \leftarrow a_1, \ldots, a_m.$" applies and $F(A_i)$ is inferred which triggers the weak constraint wc_i. The same holds for $F(B_i)$ such that wc_i applies (resp. is blocked) if and only if $(B_i|A_i)$ is applicable (resp. is verified). For further details on this procedure, we refer the reader to [6]. It should be noted that the complexity of rewriting a formula into disjunctive normal form is exponential in the worst case, i.e., $\mathcal{O}(2^n)$, as the size of the resulting expression may grow exponentially relative to the the original input.

7. Conclusions and Future Work

In this paper, we considered an approach to prioritized answer set programming where beliefs about priorities among the answer sets of an ASP program $\mathcal{P}$ are made explicit by a conditional belief base Δ. A ranking function is then generated from Δ which yields a ranking of the answer sets of $\mathcal{P}$. While this setting combining two major approaches to nonmonotonic reasoning offers a rich variety of modelling options, knowledge engineering requires a good understanding of the reasoning and ranking processes in both frameworks. We pointed out crucial differences between modelling information in ASP vs. via conditionals, highlighting the different roles that $\mathcal{P}$ and Δ play. As a side effect, our investigations also provided deep insights into differences between the basic reasoning processes in ASP vs. ranking-based conditionals. As future work, we plan to evaluate our combined approach by principles which will be inspired by those presented in [3].

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project numbers 496727276 and 512363537, grant KE 1413/14-1 awarded to Gabriele Kern-Isberner, and grant BE 1700/12-1 awarded to Christoph Beierle. Alexander Hahn was supported by grant KE 1413/14-1, Andre Thevapalan and Marco Wilhelm were partially supported by grant KE 1413/14-1, and Lars-Phillip Spiegel was supported by grant BE 1700/12-1.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–386. doi:10.1007/BF03037169.

[2] V. Lifschitz, Answer Set Programming, Springer, 2019. doi:10.1007/978-3-030-24658-7.

[3] G. Brewka, T. Eiter, Preferred answer sets for extended logic programs, Artif. Intell. 109 (1999) 297–356. doi:10.1016/S0004-3702(99)00015-6.
[4] C. Sakama, K. Inoue, Prioritized logic programming and its application to commonsense reasoning, Artif. Intell. 123 (2000) 185–222. doi:10.1016/S0004-3702(00)00054-0.
[5] M. Wilhelm, A. Thevapalan, G. Kern-Isberner, Prioritizing answer sets based on conditional expert knowledge, in: M. Franklin, S. A. Chun (Eds.), Proc. 36th Int. Florida AI Research Society Conf. (FLAIRS 2023), Florida Online Journals, 2023. doi:10.32473/flairs.36.133167.
[6] M. Wilhelm, A. Thevapalan, G. Kern-Isberner, Integrated use of system Z for preferred answer set programming, in: D. A. Talbert, I. Biskri (Eds.), Proc. 38th Int. Florida AI Research Society Conf. (FLAIRS 2025), Florida Online Journals, 2025. doi:10.32473/flairs.38.1.138664.
[7] J. Pearl, System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning, in: Proc. 3rd Conf. on Theoretical aspects of reasoning about knowledge (TARK 1990), Morgan Kaufmann Publishers Inc., 1990, pp. 121–135.
[8] G. Kern-Isberner, A thorough axiomatization of a principle of conditional preservation in belief revision, Ann. Math. Artif. Intell. 40 (2004) 127–164. doi:10.1023/A:1026110129951.
[9] W. Spohn, Ordinal conditional functions: a dynamic theory of epistemic states, in: W. Harper, B. Skyrms (Eds.), Causation in Decision, Belief Change, and Statistics, II, Kluwer Academic Publishers, 1988, pp. 105–134.
[10] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003. doi:10.1017/CBO9780511543357.
[11] M. Gelfond, Chapter 7 answer sets, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, volume 3 of *Foundations of Artificial Intelligence*, Elsevier, 2008, pp. 285–316. doi:10.1016/S1574-6526(07)03007-6.
[12] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Data Bases, Plenum Press, New York, 1978, pp. 293–322. doi:10.1007/978-1-4684-3384-5_11.
[13] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV system for knowledge representation and reasoning, ACM Trans. Comput. Log. 7 (2006) 499–562. doi:10.1145/1149114.1149117.
[14] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (2019) 27–82. doi:10.1017/S1471068418000054.
[15] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, ASP-Core-2 input language format, 2012. URL: https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.0.pdf.
[16] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, T. Schaub, ASP-Core-2 input language format, Theory Pract. Log. Program. 20 (2020) 294–309. doi:10.1017/S1471068419000450.
[17] M. Goldszmidt, J. Pearl, Qualitative probabilities for default reasoning, belief revision, and causal modeling, Artif. Intell. 84 (1996) 57–112.
[18] J. Haldimann, C. Beierle, G. Kern-Isberner, T. Meyer, Reasoning with system W and infeasible worlds, Ann. Math. Artif. Intell. 93 (2025) 665–698. doi:10.1007/S10472-025-09982-W.
[19] J. Haldimann, C. Beierle, G. Kern-Isberner, Inductive inference from weakly consistent belief bases, The Knowledge Engineering Review 40 (2025) 1–32. doi:10.1017/S0269888925100088.
[20] G. Kern-Isberner, C. Beierle, G. Brewka, Syntax splitting = relevance + independence: New postulates for nonmonotonic reasoning from conditional belief bases, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proc. 17th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2020), IJCAI Organization, 2020, pp. 560–571. doi:10.24963/kr.2020/56.
[21] R. Dennison, J. Heyninck, T. Meyer, Defeasible conditionals using answer set programming, in: M. Gebser, D. Inclezan, F. Ricca, M. Carro, M. Truszczynski (Eds.), Proc. 41st Int. Conf. on Logic Programming, (ICLP 2025), EPTCS, 2025, pp. 206–223. doi:10.4204/EPTCS.439.15.
[22] M. Alviano, L. Giordano, D. Theseider Dupré, A framework for conditional reasoning in answer set programming, in: M. Gebser, D. Inclezan, F. Ricca, M. Carro, M. Truszczynski (Eds.), Proc. 41st Int. Conf. on Logic Programming (ICLP 2025), 2025, pp. 188–201. doi:10.4204/EPTCS.439.13.