Prioritizing Answer Sets Based on Conditional Expert Knowledge

Marco Wilhelm and **Andre Thevapalan** and **Gabriele Kern-Isberner**
Department of Computer Science, TU Dortmund University, Dortmund, Germany
{marco.wilhelm, andre.thevapalan}@tu-dortmund.de, gabriele.kern-isberner@cs.tu-dortmund.de

Abstract

Answer set programming (ASP) and conditional reasoning both are powerful and widely used methodologies from the field of knowledge representation and reasoning (KR) which are capable of formalizing default statements that usually hold but also leave room for exceptions. While ASP convinces with an intuitive rule-based syntax and fast solvers, conditionals come along with a sophisticated preference-based semantics. Here, we combine both approaches by calculating answer sets which we then prioritize based on conditional expert knowledge. We apply our hybrid approach to the task of planning warehouse layouts from the logistics domain which is predestinated for our approach because it involves, on the low-level, many variables and technical framework conditions (like rack positions) and, on the high-level, expert knowledge of the layout designer.

Introduction

Answer set programming (ASP, (Baral 2010; Gelfond and Lifschitz 1988; Gelfond 2008)) is a well-established knowledge representation and reasoning (KR) formalism which has found its way into many application domains (Takeuchi et al. 2023; Abels et al. 2021; Gebser et al. 2018; Schieweck, Kern-Isberner, and ten Hompel 2017). As a most important feature of ASP, *default negation* (Clark 1977) allows for reasoning with rules such as "$B \leftarrow A, \mathtt{not}\ C$." stating that if A holds, then B follows unless C is proven to be true, even if specific information on the default-negated literal C is not available. Hence, ASP implements uncertain reasoning in a purely symbolic manner. Answer set solvers like `clingo` (cf., e.g., (Gebser et al. 2007)) can be used to efficiently compute *answer sets* of ASP programs that are consistent sets of ground literals which satisfy all rules of the program. ASP programs can have multiple answer sets, corresponding to alternative solutions, but also no answer sets when the program is inconsistent and, thus, are suited to determine all feasible solutions of a constraint satisfaction problem.

In real world applications, ASP programs can lead to a vast number of answer sets. Thus, several approaches on *prioritized/preferred ASP* have emerged (Brewka and Eiter 1999; Delgrande, Schaub, and Tompits 2000; Nieuwenborgh and Vermeir 2006; Wang, Zhou, and Lin 2000; Sakama and Inoue 2000; Zhang 2018) in order to rank the answer sets of an ASP program and select the most suited one. The typical approach is to define an order over the rules (or literals) of the program which induces a preference relation on the answer sets. The problem with this is that the knowledge engineer has to define the order by hand which usually is a non-trivial black box process. In particular, the background knowledge of the engineer that was used to establish the order on the rules (or literals) remains unrepresented.

In this paper, we combine ASP with *conditional reasoning* (Nute 1980; Kraus, Lehmann, and Magidor 1990; Kern-Isberner 2001) and develop a hybrid approach with which it is possible to rank the answer sets based on conditional expert knowledge. A *(plausibilistic) conditional* $(B|A)$ is deemed to express a semantic-based relationship between its antecedent A and its consequent B which can be read as: "If A holds, then usually B holds, too." Conditionals allow for modeling soft constraints by suggesting B in the context of A without postulating it. Note that this is substantially different from the reading of ASP rules and leads to a total preorder on possible worlds. The basic idea of our approach is that the knowledge engineer represents her background knowledge in form of conditionals, infers such a total preorder on possible worlds based on a principled conditional reasoning formalism, and deduces the preference ordering on the answer sets from this total preorder. For the reasoning formalism, we rely on the well-established *c-representations* (Kern-Isberner 2004) but, in principle, every ranking-based model of conditionals can be used (e.g., based on System Z (Pearl 1990; Kern-Isberner and Beierle 2015)).

The main benefit of our approach is the principled generation of the preferences on the answer sets of an ASP program based on conditional expert knowledge. All answer sets are maintained while common approaches on prioritized ASP return the most preferred answer sets only. Further, our approach leads to a clear separation of hard constraints in form of ASP rules and soft constraints in form of conditionals. Because preferences are treated externally, any ASP solver can be used to calculate the answer sets. The same holds for the conditional reasoning part in which we can rely on well-established semantics. Finally, the conditional knowledge can be easily updated without the need to rethink the preference ordering because the reordering is computed automatically.

With ASP and ranking functions, we bring together two major approaches to nonmonotonic logics in AI. Therefore, our hybrid framework is expected to be relevant for many applications to symbolic, uncertain reasoning. Here, we present the task of planning a logistical warehouse which constitutes a predestinated application area for our hybrid approach. So far, the automation of holistic warehouse layout planning is considered hardly possible (Gudehus 2010). Usually, it is executed by interdisciplinary teams whose efficiency suffers from unclear and conflicting goals and planners heavily rely on their own experience. Formalizing the experts' knowledge and experience in order to create computer-aided tools is essential. With our approach, we aim at providing the methodological foundations for the automation of warehouse layout planning. For the modeling process, it is particularly helpful that our approach makes it possible to separate the generally applicable technical knowledge formalized in ASP from the rather subjective and uncertain expert knowledge in form of conditionals.

The rest of the paper is organized as follows. First, we recall some basics on answer set programming and conditional reasoning based on ranking functions. After that, we discuss related work on prioritized ASP before we propose our new hybrid approach on prioritizing answer sets. Eventually, we illustrate our approach by means of an example from warehouse layout planning and conclude.

Preliminaries

As a base language, we consider a *function free first-order language* $\mathcal{FOL}$ which is defined over finite sets of predicates and constants (also known as *relational language* (Genesereth and Kao 2017)). *Atoms* in $\mathcal{FOL}$ are predicates of arity n together with n constants or variables. *Formulas* are built recursively based on the atoms in $\mathcal{FOL}$ by the use of the common connectives $(\neg, \wedge, \vee, \forall, \exists)$. A *literal* is either an atom or its negation. Formulas without free, i.e., unbounded variables are called *ground formulas* resp. *sentences*. The set of *ground atoms* is denoted by $\mathcal{G}$. Formulas with free variables can be *grounded* by substituting each free variable with a constant. We sometimes indicate negations by an overline and conjunctions by juxtaposition.

Answer Set Programming. In *answer set programming* (ASP, (Baral 2010; Gelfond 2008)) one considers *extended logic programs* (ELPs) which are finite sets of *rules*

$$r: \quad h \leftarrow b_1, \ldots, b_k, \mathsf{not}\ b_{k+1}, \ldots, \mathsf{not}\ b_m.,$$

where h and $b_1, \ldots, b_m$ are literals. With the concept of *default negation*, indicated by "not" (also known as *negation as failure* or *weak negation*, (Clark 1977)), it is possible to formulate default statements in ASP which reflect general behavior but allow for exceptions. The intuitive meaning of the rule r is: "h follows from $b_1, \ldots, b_k$ unless any $b_{k+1}, \ldots, b_m$ is provably true."

The formal semantics of ELPs is given by *answer sets*. An *answer set* $\mathcal{A}$ of an ELP $\mathcal{P}$ in which all literals are grounded is a consistent set of ground literals such that for each rule $r \in \mathcal{P}$, if $b_1, \ldots, b_k \in \mathcal{A}$ and $b_{k+1}, \ldots, b_m \notin \mathcal{A}$ hold, then $h \in \mathcal{A}$ holds, too, and $\mathcal{A}$ is minimal with this property. While classical logic programs have a unique solution, ground ELPs may have several answer sets or none. With $\mathsf{AS}(\mathcal{P})$ we denote the set of all answer sets of $\mathcal{P}$. Non-ground ELPs are grounded before calculating their answer sets by substituting the variables with each constant once.

Conditionals and Ranking Functions. In conditional logics (Nute 1980; Kraus, Lehmann, and Magidor 1990; Kern-Isberner 2001), the base language, here $\mathcal{FOL}$, is extended by a conditional operator $|$ which combines two formulas $A, B \in \mathcal{FOL}$ to a conditional expression $(B|A)$ with the intuitive meaning: "If A holds, then usually B follows." Finite sets of conditionals are called (conditional) *knowledge base*. Note that also plausible statements can be part of a knowledge base, being represented as a conditional with tautological antecedent $(A|\top)$ with $A \in \mathcal{FOL}$.

Similar to ASP rules, conditionals represent default statements but differ in their formal semantics. Here, conditionals are interpreted by *ranking functions* (Spohn 2014) which provide a preference order on the set of *possible worlds*. A *possible world* ω is a complete conjunction of ground literals, i.e., each ground atom occurs in ω once either negated or positive. A *ranking function* is a mapping $\kappa\colon \Omega \to \mathbb{N}_0^\infty$ from the set of all possible worlds Ω to the natural numbers (including 0 and ∞) such that $\kappa^{-1}(0) \neq \emptyset$. The lower the rank of a world, the more plausible the world is. The *rank* of a sentence A is $\kappa(A) = \min_{\omega \in \Omega\colon\, \omega \models A} \kappa(\omega)$. If A, B are sentences, then κ *models* the conditional $(B|A)$, in symbols

$$\kappa \models (B|A), \quad \text{iff} \quad \kappa(AB) < \kappa(A\overline{B}) \quad \text{or} \quad \kappa(A) = \infty. \quad (1)$$

Eventually, a ranking function κ is a *ranking model* of a knowledge base Δ iff κ models every conditional in Δ.

The semantics of *open conditionals* $(B|A)$ where A or B mentions free variables is not straightforward because there are many ways of defining suitable ranking models. In this paper, we make use of the approach presented in (Kern-Isberner and Thimm 2012) that is based on so-called *representatives* ("prototypes") which establish conditional relationships for open conditionals. Representatives, or representative vectors of constants, respectively, are not given externally but can be elaborated from the conditional relationships specified in the knowledge base by internal calculations. They serve as "most typical" objects for an open conditional. For further details, please see (Kern-Isberner and Thimm 2012). In our examples presented in this paper, all objects may serve as representatives. Therefore, instead of applying the elaborated semantics of open conditionals, we can ground all conditionals in a knowledge base first and interpret the resulting conditionals via (1), similar to the grounding process in ASP.

c-Representations. A knowledge base is *consistent* iff it has at least one ranking model. As $\mathbb{N}$ and, therewith, potential κ-ranks are unbounded, a consistent knowledge base automatically has infinitely many models and the selection of a specific model is crucial for inductive reasoning. In this paper, we focus on the so-called *c-representations* as ranking models (Kern-Isberner 2004; Kern-Isberner and Thimm 2012) but our hybrid approach which we will present in the next section works with any other ranking model in the same way. A ranking model κ_c^Δ of a consistent knowledge base

$\Delta = \{\delta_1, \ldots, \delta_n\}$ is a *c-representation* of Δ iff there are $\eta_i \in \mathbb{N}_0$ for $i = 1, \ldots, n$ and a normalization constant η_0 which ensures $(\kappa_c^\Delta)^{-1}(0) \neq \emptyset$ such that, for $\omega \in \Omega$,

$$\kappa_c^\Delta(\omega) = \eta_0 + \sum_{i=1,\ldots,n} \sum_{(B|A) \in \mathsf{grnd}(\delta_i):\ \omega \models A\overline{B}} \eta_i,$$

where $\mathsf{grnd}(\delta_i)$ is the set of proper groundings of δ_i. The value η_i can be understood as a *penalty point* for falsifying the i-th conditional in Δ.

Marginalization. For our hybrid approach, the concept of *marginalization* is essential. The *marginalization* of an answer set $\mathcal{A}$ on a set of ground atoms $\mathcal{G}' \subseteq \mathcal{G}$ is defined by

$$\mathcal{A}_{|\mathcal{G}'} = \{a \mid a \in \mathcal{A} \cap \mathcal{G}'\} \cup \{\neg a \mid \neg a \in \mathcal{A}, a \in \mathcal{G}'\}.$$

The *marginalization* of a ranking function κ on $\mathcal{G}' \subseteq \mathcal{G}$ is

$$\kappa_{|\mathcal{G}'}(\omega_{\mathcal{G}'}) = \min_{\omega' \in \Omega:\ \omega' \models \omega_{\mathcal{G}'}} \kappa(\omega'), \qquad \omega_{\mathcal{G}'} \in \Omega_{\mathcal{G}'},$$

where $\Omega_{\mathcal{G}'}$ is the set of (partial) possible worlds defined over $\mathcal{G}'$. That is, $\omega_{\mathcal{G}'} \in \Omega_{\mathcal{G}'}$ mentions ground atoms from $\mathcal{G}'$ only. Marginalization on $\mathcal{G}'$ means a projection on $\mathcal{G}'$.

Related Work on Prioritized ASP

Modeling real world problems with ASP can lead to a vast number of answer sets which are not necessarily equally meaningful/useful. This is particularly the case when "soft constraints" matter their intention is not to constrain the answer sets but to imply preferences among them. Soft constraints can hardly be formalized in ASP. Thus, several approaches on *prioritized/preferred ASP* have emerged (Brewka and Eiter 1999; Delgrande, Schaub, and Tompits 2000; Nieuwenborgh and Vermeir 2006; Wang, Zhou, and Lin 2000; Sakama and Inoue 2000; Zhang 2018), with the software system `asprin` (Brewka et al. 2015) leading the way. In (Brewka and Eiter 1999), a partial order over the rules of an ELP $\mathcal{P}$ is specified which describes preferences among these rules. For instance, the program

$$\begin{aligned}\mathcal{P} = \{&r_1\colon \mathsf{penguin} \leftarrow .,\ \ r_2\colon \mathsf{bird} \leftarrow .,\\ &r_3\colon \neg\mathsf{flies} \leftarrow \mathsf{penguin}, \mathsf{not\ flies}.,\\ &r_4\colon \mathsf{flies} \leftarrow \mathsf{bird}, \mathsf{not}\ \neg\mathsf{flies}. \qquad \}\end{aligned} \tag{2}$$

is given as an example. The program $\mathcal{P}$ has the answer sets

$$\mathsf{AS}(\mathcal{P}) = \{\{\mathsf{penguin}, \mathsf{bird}, \neg\mathsf{flies}\}, \{\mathsf{penguin}, \mathsf{bird}, \mathsf{flies}\}\}$$

which emerge because based on the facts r_1 and r_2 both rules r_3 and r_4 are applicable but they are exclusive. If the order of the rules in $\mathcal{P}$ is understood as a preference order, one can argue that the answer set $\{\mathsf{penguin}, \mathsf{bird}, \neg\mathsf{flies}\}$ has to be preferred over the other. Therewith, the conflict that a penguin inherits the ability to fly from the superclass of birds while penguins actually are not able to fly is solved.

One problem with this approach is that the order in which the ASP rules should be applied is not always as clear as it is in this example and the knowledge engineer has to consider the ELP as a whole when giving preferences to the rules.

The approaches in (Wang, Zhou, and Lin 2000; Delgrande, Schaub, and Tompits 2000; Nieuwenborgh and Vermeir 2006) go in the same direction as (Brewka and Eiter 1999; Brewka et al. 2015) but differ in how strict the preferences are understood. In (Sakama and Inoue 2000) the priorities are specified among the literals of the program. Different to that, in (Zhang 2018) additional subjective literals are introduced in order to represent introspection of preferences over propositions. Another widely used methodology to prioritize answer sets is optimization (Gebser, Kaminski, and Schaub 2011). For example, `smodels` (Simons, Niemelä, and Soininen 2002) provides optimization statements for expressing cost functions on sets of weighted literals. But all these approaches have in common that the preferences have to be specified explicitly by the knowledge engineer.

Input:	ELP $\mathcal{P}$ and consistent knowledge base Δ
Output:	Preference ordering on $\mathsf{AS}(\mathcal{P})$
1	Calculate answer sets of $\mathcal{P}$
2	Calculate a ranking model κ of Δ
3	Determine shared ground atoms $\mathcal{G}_\cap$
4	Marginalize κ and all $\mathcal{A} \in \mathsf{AS}(\mathcal{P})$ on $\mathcal{G}_\cap$
5	Calculate $\kappa_{\mid\mathcal{G}_\cap}(\mathcal{A}_{\mid\mathcal{G}_\cap})$ for all $\mathcal{A} \in \mathsf{AS}(\mathcal{P})$
6	Determine ordering $\leq_\kappa$ on $\mathsf{AS}(\mathcal{P})$ induced by $\leq_\kappa$
7	Return ordered partition of $\mathsf{AS}(\mathcal{P})$ induced by $\leq_\kappa$

Figure 1: Algorithm for prioritizing answer sets based on conditional expert knowledge.

Prioritization of Answer Sets Based on Conditional Knowledge

Here, we present a hybrid approach based on conditional reasoning and ASP which uses conditional knowledge to prioritize answer sets. With conditionals, we can make the background knowledge explicit which motivates the knowledge engineer to assign the preferences in prioritized ASP as they are. The main advantage of our approach is that the conditionals can be formulated separately and the global preference ordering is inductively inferred automatically.

For our approach, we consider a pair $(\mathcal{P}, \Delta)$ consisting of an ELP $\mathcal{P}$ and a consistent conditional knowledge base Δ. Let $\mathcal{FOL}_\mathcal{P}$ be the base language induced by $\mathcal{P}$ with its set of ground atoms $\mathcal{G}_\mathcal{P}$, and let $\mathcal{FOL}_\Delta$ be the base background language induced by Δ with its set of ground atoms $\mathcal{G}_\Delta$. It is crucial that the set of the shared ground atoms, i.e., $\mathcal{G}_\cap = \mathcal{G}_\mathcal{P} \cap \mathcal{G}_\Delta$, is non-empty.

The basic idea of our approach is to marginalize the answer sets of $\mathcal{P}$ as well as a ranking model of Δ on $\mathcal{G}_\cap$ and to assign ranks to the marginalized answer sets (cf. also Figure 1). First, we calculate the answer sets of $\mathcal{P}$ and a ranking model κ of Δ, e.g., by a suitable c-representation. Afterwards, both the answer sets and the ranking model are marginalized on the shared ground atoms, i.e., we determine $\mathcal{A}_{\mid\mathcal{G}_\cap}$ for $\mathcal{A} \in \mathsf{AS}(\mathcal{P})$ as well as $\kappa_{\mid\mathcal{G}_\cap}$. For this step, we need $\mathcal{G}_\cap \neq \emptyset$. Then, for each answer set $\mathcal{A} \in \mathsf{AS}(\mathcal{P})$, we compute the ranks by

$$\kappa_{\mid\mathcal{G}_\cap}(\mathcal{A}_{\mid\mathcal{G}_\cap}) := \kappa_{\mid\mathcal{G}_\cap}(\textstyle\bigwedge_{l \in \mathcal{A}_{\mid\mathcal{G}_\cap}} l)$$

which are well-defined because answer sets are consistent

ω	κ_c^Δ	ω	κ_c^Δ	ω	κ_c^Δ	ω	κ_c^Δ
ABC	1	$\overline{A}BC$	2	$AB\overline{C}$	3	$\overline{A}B\overline{C}$	2
$A\overline{B}C$	1	$\overline{A}\overline{B}C$	1	$A\overline{B}\overline{C}$	0	$\overline{A}\overline{B}\overline{C}$	1

Table 1: c-Representation of the sports match example wrt. $\eta_0 = 0$, $\eta_1 = \eta_3 = 1$, and $\eta_2 = 2$ (cf. Example 1).

sets of ground literals, i.e., they correspond to partial possible worlds. The computed ranks induce a total preorder on the answer sets in $\mathsf{AS}(\mathcal{P})$ by

$$\mathcal{A} \leq_\kappa \mathcal{A}' \text{ iff } \kappa_{|\mathcal{G}_\cap}(\mathcal{A}_{|\mathcal{G}_\cap}) \leq \kappa_{|\mathcal{G}_\cap}(\mathcal{A}'_{|\mathcal{G}_\cap}).$$

From this total preorder on $\mathsf{AS}(\mathcal{P})$, an ordered partition $(\mathfrak{A}_1, \ldots, \mathfrak{A}_m)$ of $\mathsf{AS}(\mathcal{P})$ can be deduced by $\mathcal{A} \in \mathfrak{A}_i$ iff

$$\forall \mathcal{A}' \in \mathfrak{A}_i \colon \mathcal{A} =_\kappa \mathcal{A}' \text{ and } \forall \mathcal{A}' \in \bigcup_{j=i+1}^{m} \mathfrak{A}_j \colon \mathcal{A} <_\kappa \mathcal{A}'$$

for $i = 1, \ldots, m$. Eventually, we prefer answer sets in $\mathfrak{A}_i$ to answer sets in $\mathfrak{A}_j$ if $i < j$.

Example 1. *Consider the ELP* $\mathcal{P} = \{r_1, r_2, r_3\}$ *with*

$$r_1 : \quad \mathsf{final} \leftarrow \mathsf{not\ canceled}.\ ,$$
$$r_2 : \quad A \leftarrow \mathsf{final}, \mathsf{not}\ B.\ ,$$
$$r_3 : \quad B \leftarrow \mathsf{final}, \mathsf{not}\ A.\ ,$$

stating that a final match of a sports event is happening as long as it is not canceled, and one of the competing teams will leave the field as a winner (either team A or team B). The answer sets of $\mathcal{P}$ *are* $\mathsf{AS}(\mathcal{P}) = \{\{A, \mathsf{final}\}, \{B, \mathsf{final}\}\}$.

Now, assume that a spectator wants to bet on the winner of the tournament. She formalizes her knowledge that team A usually beats team C, a third team in the tournament, that team C usually beats team B, and that there can be only one winner of the tournament in the knowledge base

$$\Delta = \{(A|A \vee C), (C|B \vee C), (A\overline{B}\overline{C} \vee \overline{A}B\overline{C} \vee \overline{A}\overline{B}C|\top)\}.$$

Note that no direct comparison between the teams A and B is available but plausibly, if A is better than C and C is better than B, one would expect A to beat B if nothing else is known. The spectator comes up with the c-representation κ_c^Δ *as shown in Table 1. As the set of shared ground atoms of* $\mathcal{P}$ *and* Δ *is* $\mathcal{G}_\cap = \{A, B\}$*, the marginalization of* κ_c^Δ *on* $\mathcal{G}_\cap$ *is*

$$\kappa_c^\Delta{}_{|\mathcal{G}_\cap}(AB) = 1, \qquad \kappa_c^\Delta{}_{|\mathcal{G}_\cap}(\overline{A}B) = 1,$$
$$\kappa_c^\Delta{}_{|\mathcal{G}_\cap}(A\overline{B}) = 0, \qquad \kappa_c^\Delta{}_{|\mathcal{G}_\cap}(\overline{A}\overline{B}) = 1.$$

Consequently, $\kappa_c^\Delta{}_{|\mathcal{G}_\cap}(A) = 0$ *and* $\kappa_c^\Delta{}_{|\mathcal{G}_\cap}(B) = 1$ *hold, which is why* $\{A, \mathsf{final}\} <_{\kappa_c^\Delta} \{B, \mathsf{final}\}$ *and the spectator should prefer the answer set* $\{A, \mathsf{final}\}$ *and bet on team A.*

The example from (Brewka and Eiter 1999) (cf. the section on related work) can be mimicked with our approach by the pair $(\mathcal{P}, \Delta)$ where $\mathcal{P}$ is the plain ELP (2) without any ordering on the rules and Δ consists of the conditional

$$r = (\mathsf{penguin} \wedge \neg\mathsf{flies} | \mathsf{penguin} \wedge \neg\mathsf{flies} \vee \mathsf{bird} \wedge \mathsf{flies}).$$

The conditional r causes that answer sets which mention the ground literals penguin and ¬flies are preferred over answer sets which mention bird and flies, observing specificity. The other way around, it is not clear how to handle our following example from warehouse layout planning wrt. (Brewka and Eiter 1999) because it combines preferences and optimization when preferring specific types of racks, for instance.

size of storage room	#racks	#answer sets	CPU time (in sec)
5×5	11	22,185	0.884
5×5	12	4,463	0.339
5×5	13	482	0.145
5×5	14	0	0.047
6×6	18	192,027	1,767.489
6×6	19	21,552	25.032
6×6	20	1,584	3.476
6×6	21	66	1.204
6×6	22	1	0.636
6×6	23	0	0.178
7×7	29	52	4,016.720
7×7	30	0	111.612

Table 2: Number of answer sets and runtime in seconds of the ELP in Figure 2 for different combinations of storage room sizes and numbers of racks.

Application to Warehouse Layout Planning

We demonstrate how our hybrid approach can assist logistics experts with the planning of warehouse layouts. For this, we consider the task of placing x racks of size 1×1 into a quadratic storage room of size $m \times m$ that is surrounded by walls and has a door on the left through which the room can be entered. The following two constraints shall be satisfied:

C1 Each cell of the grid which is free (i.e., neither a wall nor a rack) is reachable from the door.

C2 Each rack is accessible, i.e., there is a free cell next to at least one side of the rack.

Our ELP which solves this task is shown in Figure 2, here with the instance data $n = 8$ (n includes the walls, i.e., $m = n - 2$) and $x = 21$. The program uses some language extensions which go beyond the standard ASP syntax as described in the preliminaries and which are provided by `clingo` (Gebser, Kaminski, and Schaub 2011), the software system that we used to calculate the answer sets. For example, in line 22 (Figure 2), numberOfRacks-many racks are positioned in feasible cells (X, Y), where "feasible cells" means cells in accordance with **C1** and **C2**.

The number of answer sets (= produced layouts) depends on the problem instance (cf. Table 2). If $m = 6$ and $x = 21$, then there are 66 feasible layouts. Four of them are shown in Figure 3. Not all calculated layouts are equally suited for realization. For example, a warehouse designer might prefer

R1 racks which stand back to back in a block above racks in a row because of the deeper storage area that is formed (cf. the racks in the middle vs. the racks on the left and right in layout B (Figure 3)),

```
 1  % Generate problem instance (number of racks and grid cells)
2-4 #const numberOfRacks = 21.      #const n = 8.      cell(1..n, 1..n).

 5  % Place surrounding walls with door at (1, 4)
6-10 wall(1, 1..3).      wall(1, 5..n).      wall(n, 1..n).      wall(1..n, (1; n)).      door(1, 4).

11  % By default, cells are free
12        free(X, Y) ← cell(X, Y), not rack(X, Y), not wall(X, Y), not door(X, Y).

13  % Each free cell must be reachable from the door
14   reachable(X, Y) ← door(X, Y).
15   reachable(X, Y) ← free(X, Y), reachable(X, (Y − 1; Y + 1)).
16   reachable(X, Y) ← free(X, Y), reachable((X − 1; X + 1), Y).
17 ¬reachable(X, Y) ← free(X, Y), not reachable(X, Y).
18                   ← ¬reachable(X, Y).

19  % Place racks at feasible positions, i.e., next to free cells
20     feasible(X, Y) ← cell(X, Y), free(X, (Y − 1; Y + 1)), not wall(X, Y), not door(X, Y).
21     feasible(X, Y) ← cell(X, Y), free((X − 1; X + 1), Y), not wall(X, Y), not door(X, Y).
22 {rack(X, Y) : feasible(X, Y)} = numberOfRacks.
```

Figure 2: ELP of the warehouse layout planning example.

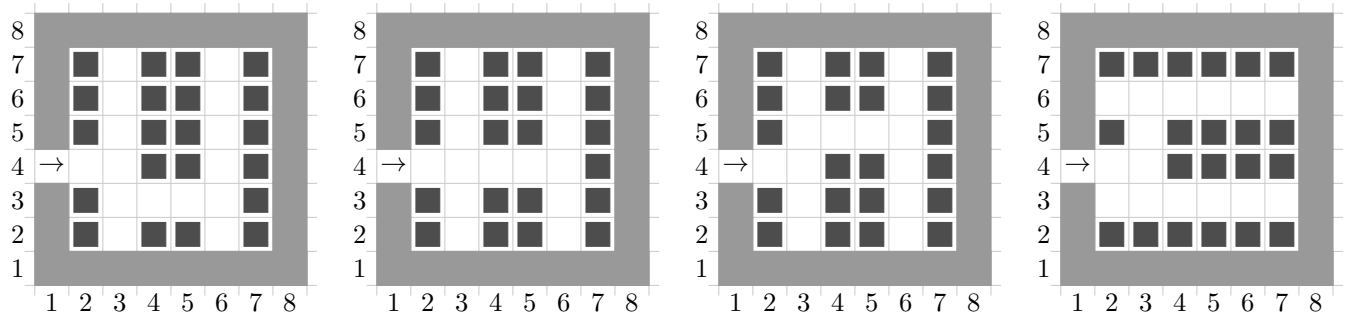

Figure 3: Some of the 66 storage room layouts produced by the ELP in Figure 2.

```
1...22: see Figure 2

23  % Classify racks that stand alone as single racks
24   singleRack(X, Y) ← rack(X, Y), not rack(X, Y − 1), not rack(X, Y + 1), not rack(X − 1, Y), not rack(X + 1, Y).

25  % Classify racks that stand in a block of at least four racks as racks in a block
26 rackInBlock(X, Y) ← rack(X, Y), rack(X, Y + 1), rack(X + 1, Y), rack(X + 1, Y + 1).
27 rackInBlock(X, Y) ← rack(X, Y), rack(X, Y + 1), rack(X − 1, Y), rack(X − 1, Y + 1).
28 rackInBlock(X, Y) ← rack(X, Y), rack(X, Y − 1), rack(X + 1, Y), rack(X + 1, Y − 1).
29 rackInBlock(X, Y) ← rack(X, Y), rack(X − 1, Y), rack(X, Y − 1), rack(X − 1, Y − 1).

30  % Classify all remaining racks as racks in a row
31   rackInRow(X, Y) ← rack(X, Y), not singleRack(X, Y), not rackInBlock(X, Y).

32  % Marginalize answer sets to atoms wrt. stated predicates
33 #show singleRack/2.
34 #show rackInBlock/2.
35 #show rackInRow/2.
```

Figure 4: ELP of the warehouse layout planning example extended by the classification of racks.

ω	$\kappa_c^{\Delta^{ij}}$	ω	$\kappa_c^{\Delta^{ij}}$	ω	$\kappa_c^{\Delta^{ij}}$	ω	$\kappa_c^{\Delta^{ij}}$
brs	1	$\bar{b}rs$	3	$br\bar{s}$	1	$\bar{b}r\bar{s}$	1
$b\bar{r}s$	2	$\bar{b}\bar{r}s$	2	$b\bar{r}\bar{s}$	0	$\bar{b}\bar{r}\bar{s}$	0

Table 3: c-Representation of Δ^{ij} from the warehouse layout planning example wrt. $\eta_0 = 0$ and $\eta_1 = \ldots = \eta_4 = 1$.

R2 racks in a row above racks which stand alone scattered over the room (like the rack in cell $(2, 5)$ in layout D (Figure 3)) for stability reasons.

This kind of soft constraints shall be considered whenever possible but shall not lead to a complete rejection of layouts, which is in contrast to the meaning of the constraints **C1** and **C2**. The classification of racks as needed for **R1** and **R2** is realized in ASP (cf. the extension of our ELP in Figure 4). For example, the ASP rule

$$\begin{aligned}\mathsf{singleRack}(X,Y) \leftarrow \mathsf{rack}(X,Y), \mathsf{not\ rack}(X,Y-1),\\ \mathsf{not\ rack}(X,Y+1), \mathsf{not\ rack}(X-1,Y),\\ \mathsf{not\ rack}(X+1,Y).\end{aligned}$$

identifies racks which stand alone, i.e., which are not surrounded by any other rack. To evaluate the layouts, the classification of the racks (s = single rack, r = rack in a row, b = rack in a block) is extracted from the answer sets, i.e., the answer sets are marginalized on the ground atoms $\mathsf{singleRack}(i, j)$, $\mathsf{rackInRow}(i, j)$, and $\mathsf{rackInBlock}(i, j)$ for $i, j \in \{1, \ldots, m\}$ (cf. lines 32-35 in Figure 4) and are ranked based on a conditional knowledge base which formalizes the soft constraints **R1** and **R2**. This formalization of **R1** and **R2** is realized by $\Delta = \{\delta_1, \ldots, \delta_4\}$ with

$$\begin{aligned}\delta_1 &= (r(X,Y)|s(X,Y) \vee r(X,Y)),\\ \delta_2 &= (b(X,Y)|s(X,Y) \vee b(X,Y)),\\ \delta_3 &= (b(X,Y)|r(X,Y) \vee b(X,Y)),\\ \delta_4 &= (b(X,Y)\overline{s(X,Y)}\,\overline{r(X,Y)}\\ &\quad \vee \overline{b(X,Y)}s(X,Y)\overline{r(X,Y)}\\ &\quad \vee \overline{b(X,Y)}\,\overline{s(X,Y)}r(X,Y)\\ &\quad \vee \overline{b(X,Y)}\,\overline{s(X,Y)}\,\overline{r(X,Y)}|\top).\end{aligned}$$

Basically, δ_1 formalizes **R2** and δ_3 formalizes **R1**. Conditional δ_2 reflects the (uncertain) consequence of **R1** and **R2** that racks in a block are preferred above single racks, and δ_4 states that at each position at most one classification can apply. For the details on how a c-representation of Δ is calculated, we refer to (Kern-Isberner and Thimm 2012). Here, we just note that the conditionals can be evaluated independently per cell thanks to the concept of *syntax splitting* (Parikh 1999; Kern-Isberner, Beierle, and Brewka 2020). With Δ^{ij} we denote Δ where the variables X and Y are substituted by the constants referring to the cell (i, j). A c-representation of Δ^{ij} is shown in Table 3 where the abbreviations $s = \mathsf{singleRack}(i, j)$ as well as $r = \mathsf{rackInRow}(i, j)$ and $b = \mathsf{rackInBlock}(i, j)$ apply. The listed ranks can be understood as penalty points and, e.g.,

$$\kappa_c^{\Delta^{ij}}(b\bar{r}\bar{s}) = 0 < 1 = \kappa_c^{\Delta^{ij}}(\bar{b}r\bar{s})$$

can be interpreted as: "In cell (i, j), a rack in a block is preferred above a rack in a row." A c-representation of the complete knowledge base Δ is given then by

$$\kappa_c^{\Delta}(\omega) = \sum\nolimits_{(i,j)\in\{1,\ldots,8\}^2} \kappa_c^{\Delta^{ij}}(\omega_{ij}),$$

where ω_{ij} is the marginalization of ω on the ground atoms referring to cell (i, j). Now, we can calculate the ranks of the layouts in Figure 3,

$$\kappa_c^{\Delta}(\text{layout A}) = 13, \qquad \kappa_c^{\Delta}(\text{layout C}) = 11,$$
$$\kappa_c^{\Delta}(\text{layout B}) = 11, \qquad \kappa_c^{\Delta}(\text{layout D}) = 14,$$

by penalizing each rack in a row with 1 and each single rack with 2 (cf. Table 3, $\kappa_c^{\Delta^{ij}}(\bar{b}r\bar{s}) = 1$ and $\kappa_c^{\Delta^{ij}}(\bar{b}\bar{r}s) = 2$). For example, layout D has rank 14 because it involves twelve racks in a row and one single rack. The preferred layouts among these four layouts are layout B and C with no single racks and the lowest number of racks in a row. Hence, our approach provides the logistics expert with a list of all feasible layouts ordered with respect to the user's preferences.

Conclusion

We proposed a hybrid approach which combines answer set programming (ASP) and conditional reasoning with the goal of prioritizing answer sets based on conditional expert knowledge. Given a pair $(\mathcal{P}, \Delta)$ consisting of an extended logic program $\mathcal{P}$ and a conditional knowledge base Δ, we calculated the answer sets of $\mathcal{P}$ and a ranking model of Δ. Afterwards, we marginalized both on the intersection of the base languages of $\mathcal{P}$ and Δ and ranked the marginalized answer sets according to the marginalized ranking model. Therewith, we obtained a preorder on the answer sets of $\mathcal{P}$ which we used to prioritize them.

Our approach is motivated by the necessity of combining large constraint solving problems on the one hand and highly condensed expert knowledge on the other when planning warehouses in logistics. We proved in a small case study that ASP is capable of generating feasible warehouse layouts and that our hybrid approach can then be used to rank these layouts. As we follow a rule-based approach, the problem formulation in our example can be transferred to larger warehouse planning tasks easily. Therewith, our approach constitutes a first step towards automatizing the up to now mostly manually handled task of warehouse layout planning.

In future work, we want to embed our hybrid approach into a larger software system with the goal of handling real world problems from warehouse layout planning. On the theoretical side, we want to postulate quality criteria under which we investigate the interactions between the first-order semantics of conditionals from (Kern-Isberner and Thimm 2012) and ASP in more detail, and we also want to consider other approaches to first-order semantics of conditionals like System Z (Pearl 1990; Kern-Isberner and Beierle 2015).

Acknowledgments. This work was supported by the Research Grant KE 1413/14-1 of the German Research Foundation (DFG) awarded to Gabriele Kern-Isberner.

References

Abels, D.; Jordi, J.; Ostrowski, M.; Schaub, T.; Toletti, A.; and Wanko, P. 2021. Train scheduling with hybrid answer set programming. *Theory Pract. Log. Program.* 21(3):317–347.

Baral, C. 2010. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Brewka, G., and Eiter, T. 1999. Preferred answer sets for extended logic programs. *Artificial Intelligence* 109(1-2):297–356.

Brewka, G.; Delgrande, J.; Romero, J.; and Schaub, T. 2015. asprin: Customizing answer set preferences without a headache. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, 1467–1474. AAAI Press.

Clark, K. 1977. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, Advances in Data Base Theory, 293–322. New York: Plemum Press.

Delgrande, J.; Schaub, T.; and Tompits, H. 2000. Logic programs with compiled preferences. In Horn, W., ed., *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000*, 464–468. IOS Press.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. clasp : A conflict-driven answer set solver. In Baral, C.; Brewka, G.; and Schlipf, J., eds., *Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings*, volume 4483 of *LNCS*, 260–265. Springer.

Gebser, M.; Obermeier, P.; Schaub, T.; Ratsch-Heitmann, M.; and Runge, M. 2018. Routing driverless transport vehicles in car assembly with answer set programming. *Theory Pract. Log. Program.* 18(3-4):520–534.

Gebser, M.; Kaminski, R.; and Schaub, T. 2011. Complex optimization in answer set programming. *Theory and Practice of Logic Programming* 11(4-5):821–839.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, 1070–1080. MIT Press.

Gelfond, M. 2008. Answer sets. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier. 285–316.

Genesereth, M., and Kao, E. J. 2017. *Relational Logic*. Springer. 63–81.

Gudehus, T. 2010. *Logistik: Grundlagen, Strategien, Anwendungen*. Springer, 4 edition. In German.

Kern-Isberner, G., and Beierle, C. 2015. A system z-like approach for first-order default reasoning. In Eiter, T.; Strass, H.; Truszczynski, M.; and Woltran, S., eds., *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, number 9060 in LNAI. Springer. 81–95.

Kern-Isberner, G., and Thimm, M. 2012. A ranking semantics for first-order conditionals. In Raedt, L. D.; Bessiere, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P., eds., *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 456–461. IOS Press.

Kern-Isberner, G.; Beierle, C.; and Brewka, G. 2020. Syntax splitting = relevance + independence: New postulates for nonmonotonic reasoning from conditional belief bases. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, 560–571.

Kern-Isberner, G. 2001. *Conditionals in Nonmonotonic Reasoning and Belief Revision*, volume 2087 of *LNCS*. Springer.

Kern-Isberner, G. 2004. A thorough axiomatization of a principle of conditional preservation in belief revision. *Annals of Mathematics and Artificial Intelligence* 40(1-2):127–164.

Kraus, S.; Lehmann, D.; and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44(1-2):167–207.

Nieuwenborgh, D. V., and Vermeir, D. 2006. Preferred answer sets for ordered logic programs. *Theory and Practice of Logic Programming* 6(1-2):107–167.

Nute, D. 1980. *Topics in Conditional Logic*. Philosophical Studies Series. Springer.

Parikh, R. 1999. *Beliefs, Belief Revision, and Splitting Languages*. Center for the Study of Language and Information. 266–278.

Pearl, J. 1990. System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In Parikh, R., ed., *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge, Pacific Grove, CA, USA, March 1990*, 121–135. Morgan Kaufmann.

Sakama, C., and Inoue, K. 2000. Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence* 123(1-2):185–222.

Schieweck, S.; Kern-Isberner, G.; and ten Hompel, M. 2017. Cellular transport systems improved: Achieving efficient operations with answer set programming. In Sabourin, C.; Guervós, J. J. M.; Madani, K.; and Warwick, K., eds., *Computational Intelligence - 9th International Joint Conference, IJCCI 2017 Funchal-Madeira, Portugal, November 1-3, 2017 Revised Selected Papers*, volume 829 of *Studies in Computational Intelligence*, 305–326. Springer.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138(1-2):181–234.

Spohn, W. 2014. *The Laws of Belief - Ranking Theory and Its Philosophical Applications*. Oxford University Press.

Takeuchi, R.; Banbara, M.; Tamura, N.; and Schaub, T. 2023. Solving vehicle equipment specification problems with answer set programming. In Hanus, M., and Inclezan, D., eds., *Practical Aspects of Declarative Languages - 25th International Symposium, PADL 2023, Boston, MA, USA, January 16-17, 2023, Proceedings*, volume 13880 of *Lecture Notes in Computer Science*, 232–249. Springer.

Wang, K.; Zhou, L.; and Lin, F. 2000. Alternating fixpoint theory for logic programs with priority. In Lloyd, J.; Dahl, V.; Furbach, U.; Kerber, M.; Lau, K.-K.; Palamidessi, C.; Pereira, L. M.; Sagiv, Y.; and Stuckey, P., eds., *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, volume 1861 of *LNCS*, 164–178. Springer.

Zhang, Z. 2018. Introspecting preferences in answer set programming. In Palù, A. D.; Tarau, P.; Saeedloei, N.; and Fodor, P., eds., *Technical Communications of the 34th International Conference on Logic Programming, ICLP 2018, July 14-17, 2018, Oxford, United Kingdom*, volume 64 of *OASIcs*, 3:1–3:13. Dagstuhl.