Accepting the natural order of rules in a logic program with preferences

Alexander Šimko

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University
Mlynská dolina, 824 48 Bratislava, Slovakia
simko@fmph.uniba.sk

Abstract

Preference is a natural part of common sense reasoning. It allows us to select preferred conclusions from broader range of alternative conclusions. It is typically specified on parts of conclusions or on rules. Different semantics have been proposed that deal with preference on rules. None fully meets our requirements.

We are interested in a descriptive approach to preference handling in logic programs under answer set semantics that always selects preferred answer set when standard one exists. Existing semantics that meet this criterion also give non intuitive conclusions on some programs. We think this kind of problem is related to the problem of not accepting natural order of rules induced by underlying answer set semantics.

Our goal is to define semantics that would always select preferred answer set when standard one exists, accept natural order on rules, and satisfy principles for preference handling.

1998 ACM Subject Classification I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases non-monotonic reasoning, knowledge representation, answer set semantics, preference handling, preferred answer set

Digital Object Identifier 10.4230/LIPIcs.ICLP.2011.284

1 Introduction and problem description

In common sense reasoning some form of preference is usually used. One can prefer some conclusion over another, e.g., doctors and patients tend to prefer non invasive procedures over invasive ones. Preferences are also used to solve conflicts among rules. Having two applicable rules with contradictory effects we want to apply only preferred one.

In the last two decades logic programming has emerged as a favourite framework for knowledge representation. Especially answer set semantics of logic programming is widely used. Logic program consists of rules of the form "If a is true, and there is no evidence that b is true then also c is true". In such a program, answer set semantics gives us answer sets – sets of alternative conclusions. Existence of multiple answer sets is due to the use of default negation.

Natural questions arise: **1.** How to encode preference in logic program? **2.** How to extend answer set semantics in presence of preference?

These two questions have already been explored in literature. In next section we give an overview of existing approaches and identify places for improvement. We mainly focus on approaches that deal with preference on rules.

Technical Communications of the 27th International Conference on Logic Programming (ICLP'11).
Editors: John P. Gallagher, Michael Gelfond; pp. 284–289

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Background and overview of the existing literature

As mentioned earlier, one can consider preference on literals (e.g., [7], [2], [6]). Another option is to consider preference on rules (e.g., [1], [3], [14], [12]). [7] also provides a way for handling preferences on rules – via transformation to preference on literals.

Meaning of a logic program without preferences is a set of answer sets. Similarly, meaning of a logic program with preferences is a set of preferred answer sets. Difference between an answer set and a preferred answer set is only that in later one preferences are considered.

In a selective approach, we select some standard answer sets to be preferred answer sets [8]. We only pick from existing answer sets and do not generate new ones. In a non selective approach a preferred answer set does not have to be a standard answer set. We see such a thing as a step outside of the answer set semantics. For the purpose of this summary we restrict our focus only to selective approaches.

Approaches that deal with preference on rules are traditionally divided into two groups: prescriptive and descriptive. Prescriptive approaches (e.g., [1], [3], [14]) view preference on rules as an order in which rules are to be applied in process of generating answer set. Descriptive approaches (e.g., [15], [7], [10], [12]) do not follow this view. Instead, they see preferences as a wish list one tries to satisfy [4].

Principles for preference handling relate a preference on rules with a preference on answer sets. First two principles were proposed in [1]. [1] also considers additional condition for preference handling: If a program has a standard answer set then it also has a preferred one. If we assume this condition the second principle is violated [1]. This additional condition is not satisfied in semantics of [1], [3], [14]. [1] also propose a relaxation of their approach, that always yields a preferred answer set when a standard one exists. But this approach satisfies none of the principles from [1].

A problem with existence of a preferred answer set is related to the view prescriptive approaches adapt. They see preference as an order in which rules are to be applied. But the answer set semantics already induces an order on rules. It is well known that stratified program induces the natural order in which rules must to be applied [9]. But the order specified by preference on rules can be in conflict with the natural order. In such case, prescriptive approaches can have difficulties to select a preferred answer sets.

[7] deals with a preference on literals. It also provides a way one can transfer a preference on rules to a preference on literals. Use of this transformation leads to the comparison of generating sets of answer sets. But the transformation is unable to track blocking between rules. If head of a rule r_1 is default negated in the body of a rule r_2 we say that r_1 blocks r_2. As shown in [11] concept of blocking is important when we compare generating sets. Preference alone is not sufficient.

[10] uses preference on rules to select "best" extended answer set. Selection of a preferred extended answer set is done by comparing program reducts – set of rules. When the comparison is made, only preference on rules is used . This semantics does not satisfy the first principle from [1]. It is also on the edge of our focus as it introduces preference in a modified semantics (extended answer set semantics).

[15] always selects a preferred answer set when a standard one exists [1]. On the other hand it gives some results that we consider counterintuitive:

▶ **Example 1.** Consider the program from Example 6 from [15].

$r_1 : p \leftarrow not\ q, not\ r$
$r_2 : q \leftarrow not\ p$
$r_3 : r \leftarrow not\ p$ $\qquad\qquad$ r_1 is preferred over r_2

It has two answer sets, $A_1 = \{p\}$ and $A_2 = \{q, r\}$, generated by $R_1 = \{r_1\}$ and $R_2 = \{r_2, r_3\}$, respectively. According to [15] both A_1 and A_2 are preferred.

We argue that only A_1 should be preferred. $r_1 \in R_1$ blocks both $r_2, r_3 \in R_2$. Also every $r \in R_2$ blocks r_1. But r_1 is preferred over r_2 and on its own generates A_1. R_2 contains one less preferred rule and no preferred one.

3 Goal of research

We also do not adopt the view of prescriptive approaches. It is well known that a stratified program induces a natural order on rules [9]. It tells us which rule must be applied before another. Moreover, every non stratified program is transformed to stratified one during the Gelfond-Lifschitz transformation – the guess phase of the answer set semantics. Rules that are filtered out in this transformation are not applicable in an answer set candidate. Note that rules, which pass this transformation for a given answer set, form a set of generating rules.

We understand this in the following way. There is no need to consider order on rules from the same generating set of an answer set. Such order is already defined. We know that every rule from generating set is applicable, and none of default negated literals in the body of rule is derivable (in a corresponding answer set). A rule can only be applied after we have derived its prerequisites. Hence, rules deriving prerequisites must be applied first. On the other hand, rules not being part of any generating set will never be applied. "Order in which they will be applied" has no meaning. Similarly, order of application between rules from different generating sets has no meaning. When we are generating answer set, we work with one generating set.

There is also another view. In a selective approach, a set of preferred answer sets is a subset of a set of standard answer sets. When we accept the principle that there must be a preferred answer set when a standard one exists and a program has only one answer set, it clearly must be a preferred one. In such a program there is no need to consider preference on rules since there is only one candidate we can choose from. Preference turns to be interesting if we have rules that produce alternative conclusions – multiple answer sets.

In accordance with this we see another interpretation of preference. We understand preference "r_2 is preferred over r_1" as follows. When rules r_1 and r_2 lead to alternative conclusions (answer sets), prefer one that uses rule r_2. But when rules participate on same conclusion or one of them is inapplicable at all (in every answer set), preference does not matter.

Condition "when rules lead to alternative conclusions" fits well into the concept of generating sets. Rules from the same generating set produce the same answer set. Different answer sets are represented by different generating sets. Same answer sets can also be represented by different generating sets. It does not cause any complication. In fact, it enables us to reason about alternative derivations. That is why we understand selection of preferred answer sets as a comparison of generating sets.

As shown in [11], preference alone is not sufficient to compare generating sets. Blocking between rules is important to determine which preference is more important.

Goal of our current research is to develop a descriptive approach to preference handling in logic programming that would: **1.** handle preference on rules, **2.** be selective, **3.** always give a preferred answer set when a standard one exists, **4.** accept natural order in which rules must to be applied, **5.** satisfy principles for preference handling (first principle from [1], fourth principle from [12], and sixth principle from [13]).

In a long term we also plan to provide an implementation and a detailed comparison to existing approaches.

4 Preliminary results accomplished

In our previous work [12], we have tried to refine approach from [11]. We have proposed an approach to preference handling that always selects a preferred answer set. It is inspired by some form of argumentation. Rules are seen as an argumentation structures. The key point is not to consider all preferences but only those among blocking rules. Such preferences are called attacks. Next, there are nine rules to combine argumentation structures into answer sets and to derive attacks on argumentation structures. Roughly speaking, preferred answer sets are then defined in terms of attacks on answer sets. We have also proposed new principles to preference handling (based on notion of attack) that enable us to correctly solve problematic examples from literature. We submitted this work to ICLP 2011. Detailed description of this approach is technically complicated and it is beyond the scope of this paper.

In order to be able to provide implementation for [12] we needed to simplify technical aspects of our approach. We have also realized that our argumentation structures represent subsets of generating rules of an answer set. We have focused on translating attacks on argumentation structures to attacks on generating sets. In [13], we have proposed a descriptive approach to preference handling that is based on the concept of attacks on generating sets. We did not try to propose equivalent approach to the one in [12]. We have proposed similar approach that tries to be compatible, satisfies principles from [12], and is based on the same understanding of preference on rules. The main idea of our approach is that a generating set being under attack cannot generate a preferred answer set. We have also proposed a new principle that expresses our understanding of preference. Preference on rules that do not generate any answer set should not matter. We have submitted [13] to ŠVK 2011, a student science conference at our university. We have also presented it there and applied it for publication in conference proceeding.

Both our approaches share a common drawback. The way they handle mutual attacks of argumentation structures/generating sets is technically oriented. It lacks intuitive interpretation and ignores a natural order of rules in logic program.

Work on approaches [12] and [13] have helped us to better understand the connection between preference on rules and blocking of rules. It is clear that not all preferences have the same importance. Preference on blocking rules should be more important.

5 Current status of the research and expected achievements

Our current research is focused on figuring out the details around the concept of preference importance. We consider the concept of preference importance to be important for our goals: to accept natural order in which rules have to be applied, and to produce intuitive results.

We see a promising direction. Splitting [5] shows that we can consider rules in a iterative manner, similar to the one we use to compute the answer set of a stratified program. The important difference is that there exist decision points where we must decide which of mutually blocking rules we want to use. These decision points are responsible for multiple answer sets. In other words, splitting creates a decision tree of rule application. In order to decide which rule to use in a decision point we have to decide which rules to apply in all former decision points. If we do not choose a particular branch of a decision tree there is no

need to consider rules in it. Hence, preference on rules on former decision points is more important than on later ones.

Next example demonstrates the concept of preference importance and sketches solution to selection of preferred answer sets.

▶ **Example 2.** Consider the following program with one decision point:

$r_1 : a \leftarrow not\ b$ $\qquad$ $r_2 : b \leftarrow not\ a$
$r_3 : c \leftarrow a, not\ d$ $\qquad$ $r_4 : d \leftarrow b$

r_1 is preferred over r_2
r_4 is preferred over r_3

It has two answer sets, $A_1 = \{a, c\}$ and $A_2 = \{b, d\}$, generated by $R_1 = \{r_1, r_3\}$ and $R_2 = \{r_2, r_4\}$, respectively.

Splitting sequence $\langle\{a, b\}, \{a, b, d\}, \{a, b, c, d\}\rangle$ divides rules into three groups $\Pi_0 = \{r_1, r_2\}$, $\Pi_1 = \{r_4\}$ and $\Pi_2 = \{r_3\}$. Due to the literal b in the body of rule r_4, group Π_1 depends on group Π_0. So there is natural order. Rules from Π_0 must be considered before rules in Π_1, and similarly Π_1 before Π_2.

In the first place, we must settle down the question of rule application for rules from Π_0. Answer sets of Π_0 are $\{a\}$ and $\{b\}$. It means that the first (also the only) decision point is whether to use rule r_1 or r_2. Since r_1 is preferred over r_2, we select to use rule r_1. We do not use r_2, so there is no way to generate A_2. Thus, A_2 is not preferred. And since r_2 is not used, also r_4 cannot be used. Consequently, there is no need to consider preference of rule r_4 over any other rule. Preference of r_4 over r_3 would be considered only if r_1 is not preferred over r_2 and r_2 is not preferred over r_1.

To sum up, we expect to propose a descriptive approach to preference handling for extended logic programs under answer set semantics with preference on rules. Our semantics should be selective, and conceptually based on comparison of generating sets. We hope to meet Principle I, III, IV and VI (from [1], [12], [13]). More importantly, we think that the use of preference importance as described above, will allow us to accept natural order of a program, and to have a semantics with intuitive conclusions. Once the semantics is defined, we plan to provide an implementation and a comparison to existing approaches.

References

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

2 Gerhard Brewka. Logic programming with ordered disjunction. In *Proceedings of AAAI-02*, pages 100–105. AAAI Press, 2002.

3 J. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, 2003.

4 James P. Delgrande and Torsten Schaub. Expressing preferences in default logic. *Artificial Intelligence*, 123(1-2):41–87, 2000.

5 Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *Proceedings of the eleventh international conference on Logic programming*, pages 23–37, Cambridge, MA, USA, 1994. MIT Press.

6 A. Mileo and T. Schaub. Extending Ordered Disjunctions for Policy Enforcement: Preliminary report. In *The 2006 Federated Logic Conference*, page 45, 2006.

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

8 Torsten Schaub and Kewen Wang. A comparative study of logic programs with preference. In *IJCAI*, pages 597–602, 2001.

9 Torsten Schaub and Kewen Wang. A semantic framework for preference handling in answer set programming. *Theory and Practice of Logic Programming*, 3(4):39, 2003.

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

11 Ján Šefránek. Preferred answer sets supported by arguments. In *Proc. of the Workshop NMR*, 2008.

12 Ján Šefránek and Alexander Šimko. Warranted derivation of preferred answer sets, http://kedrigern.dcs.fmph.uniba.sk/reports/. Technical Report TR-2011-027, Comenius University, Faculty of Mathematics, Physics, and Informatics, 2011.

13 Alexander Šimko. Preferred answer sets - banned generating set approach. 2011.

14 Kewen Wang, Lizhu Zhou, and Fangzhen Lin. Alternating fixpoint theory for logic programs with priority. In *Computational Logic*, pages 164–178, 2000.

15 Y. Zhang and N. Foo. Answer sets for prioritized logic programs. In *Proceedings of the 1997 international symposium on Logic programming*, pages 69–83, 1997.