

A Rule-Based Approach to Specifying Preferences over Conflicting Facts and Querying Inconsistent Knowledge Bases (Extended Abstract)

Meghyn Bienvenu¹, Camille Bourgaux², Katsumi Inoue³ and Robin Jean¹

¹Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France

²DI ENS, ENS, CNRS, PSL University & Inria, Paris, France

³National Institute of Informatics, Tokyo, Japan

Abstract

This extended abstract summarizes our KR'25 paper [1], where we introduce a declarative rule-based framework for specifying and computing a priority relation between conflicting facts. As the expressed preferences may contain undesirable cycles, we consider the problem of determining when a set of preference rules always yields an acyclic relation, and we also explore a pragmatic approach that extracts an acyclic relation by applying various cycle removal techniques. As a step towards an end-to-end system for querying inconsistent knowledge bases, we present a preliminary implementation and experimental evaluation of the framework, which employs answer set programming to evaluate the preference rules, apply the desired cycle resolution techniques to obtain a priority relation, and answer queries under prioritized-repair semantics.

Keywords

inconsistency handling, preference specification, inconsistency-tolerant query answering, ontology-mediated query answering, answer set programming


1. Introduction

Inconsistency-tolerant semantics are a well-established approach to querying data inconsistent w.r.t. some constraints, both in the relational database and ontology-mediated query answering settings (cf. recent surveys [2, 3]). Such semantics typically rely on (*subset*) *repairs*, defined as maximal subsets of the data consistent w.r.t. the constraints. The most well-known, called the *AR* semantics in the KR community and corresponding to consistent query answering in the database community, considers that a Boolean query holds true if it holds in every repair. The more cautious *IAR* semantics amounts to querying the repairs intersection, and the less cautious *brave* semantics only requires that the query holds in some repair.

Since an inconsistent dataset may have a lot of repairs, several notions of preferred repairs have been proposed in the literature, to restrict the possible worlds considered to answer queries, for example by taking into account some information about the reliability of the data [4, 5, 6, 7, 8]. In particular, since its introduction by [9], the framework of *prioritized databases*, in which a (binary acyclic) *priority relation* between conflicting facts is used to define three kinds of *optimal repairs*, has attracted attention, with numerous theoretical results [10, 11, 12, 13], and an implementation [14]. However, the crucial question of obtaining the priority relation was left unaddressed, preventing the adoption of this framework in practice. Indeed, it is not realistic to expect users to manually input a binary relation between the facts.

2. Specifying Priority Relations via Rules

We propose a framework for specifying a priority relation between conflicting facts. We use preference rules to state that, when some conditions are satisfied, a fact should generally be preferred to another

 DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

 meghyn.bienvenu@u-bordeaux.fr (M. Bienvenu); camille.bourgaux@ens.fr (C. Bourgaux); inoue@nii.ac.jp (K. Inoue); robin.jean@u-bordeaux.fr (R. Jean)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

fact. The rule conditions may naturally refer to the presence (or absence) of facts in the dataset. However, typically we may also want to exploit information about the facts themselves (e.g. the date they were added), provided in metadata.

Example 1. Consider a DL knowledge base $\mathcal{K}_{\text{ex}} = (\mathcal{D}_{\text{ex}}, \mathcal{T}_{\text{ex}})$ about a university. The ontology expresses that associate and full professors (APr and FPr) are faculty members (Fac) and clerical staff workers (Cleric) are administrative staff workers (Adm). Moreover, one cannot be both an associate and a full professor, or an administrative staff worker and a faculty member.

$$\begin{aligned}\mathcal{T}_{\text{ex}} &= \{\text{APr} \sqsubseteq \text{Fac}, \text{FPr} \sqsubseteq \text{Fac}, \text{APr} \sqsubseteq \neg \text{FPr}, \exists \text{Teach} \sqsubseteq \text{Fac}, \text{Cleric} \sqsubseteq \text{Adm}, \text{Adm} \sqsubseteq \neg \text{Fac}\} \\ \mathcal{D}_{\text{ex}} &= \{\text{APr}(a), \text{FPr}(a), \text{Cleric}(a), \text{Adm}(a), \text{Teach}(a, c), \text{Adm}(b), \text{APr}(b)\}\end{aligned}$$

We associate to \mathcal{K}_{ex} a meta-database $\mathcal{M}_{\text{ex}} = (\text{id}_{\text{ex}}, \mathcal{F}_{\text{ex}})$, where id_{ex} is a function that associates an identifier to each fact of \mathcal{D}_{ex} : $\text{id}_{\text{ex}}(\text{APr}(a)) = 1$, $\text{id}_{\text{ex}}(\text{FPr}(a)) = 2$, $\text{id}_{\text{ex}}(\text{Cleric}(a)) = 3$, $\text{id}_{\text{ex}}(\text{Adm}(a)) = 4$, $\text{id}_{\text{ex}}(\text{Teach}(a, c)) = 5$, $\text{id}_{\text{ex}}(\text{Adm}(b)) = 6$, $\text{id}_{\text{ex}}(\text{APr}(b)) = 7$, and \mathcal{F}_{ex} records the year that facts have been added to the university database:

$$\{\text{Date}(1, 2014), \text{Date}(2, 2025), \text{Date}(3, 2013), <(2013, 2014), <(2013, 2025), <(2014, 2025)\}.$$

We then define the following preference rules.

$$\begin{aligned}\text{Date}(x_1, y_1) \wedge \text{Date}(x_2, y_2) \wedge <(y_2, y_1) &\rightarrow \text{pref}(x_1, x_2) \\ x_1 = \text{id}(\text{FPr}(y)) \wedge x_2 = \text{id}(\text{APr}(y)) &\rightarrow \text{pref}(x_1, x_2) \\ Y \sqsubseteq \text{Adm} \wedge Z \sqsubseteq \text{Fac} \wedge \neg(\exists z \text{Teach}(y, z)) \wedge x_1 = \text{id}(Y(y)) \wedge x_2 = \text{id}(Z(y)) &\rightarrow \text{pref}(x_1, x_2)\end{aligned}$$

The first rule states a general preference for keeping more recently added facts. The second one states if we have both $\text{FPr}(p)$ and $\text{APr}(p)$, we prefer to keep $\text{FPr}(p)$, capturing the domain knowledge that associate professors are promoted into full professors. The third rule states that if a person is declared to belong both to a subclass of Adm and a subclass of Fac, but there is no Teach-fact for the person in the dataset, then the Adm-related facts are deemed more reliable. Observe that it uses ontology axioms with variables in order to simplify rule formulation (avoiding the need to write separate rules for every pair of subclasses of Adm and Fac). These three preference rules induce the following preferences over the facts of \mathcal{D}_{ex} , designated by their identifiers:

$$\{\text{pref}(2, 1), \text{pref}(2, 3), \text{pref}(1, 3), \text{pref}(6, 7)\}.$$

Given a set of preference rules Σ , let \succ_{Σ} be the binary relation over facts obtained from the preferences over facts induced by Σ , restricted to facts that appear together in some conflict (minimal set of facts inconsistent with the logical theory \mathcal{T}). This relation may still fail to be a priority relation if it contains a cycle, as priority relations are required to be acyclic. We explore two complementary approaches to tackling this issue: identifying preference rules which are guaranteed to yield an acyclic relation, and employing different methods to extract an acyclic sub-relation from \succ_{Σ} .

Checking acyclicity of preference rules We show that, given a logical theory \mathcal{T} , the problem of deciding whether a set of preference rules Σ yields an acyclic \succ_{Σ} for every dataset and meta-database is undecidable in general. However, for DL-Lite ontologies and preference rules whose bodies are essentially conjunctive queries, this problem is in coNP.

Resolving cycles to get a priority relation Ideally the preference ruleset Σ would always yield an acyclic \succ_{Σ} , but this cannot be assumed in general. Furthermore, cycles can naturally arise when users create rules that capture different criteria, e.g. prefer more recent facts and prefer facts from more trusted sources. To ensure acyclicity in such cases, one would need to create more complex rules whose bodies consider different combinations of the criteria, making rules much harder for users to specify and understand. We thus advocate a pragmatic approach: give users free rein to specify preferences as they see fit, then apply cycle resolution techniques to extract a suitable acyclic sub-relation should any cycles arise. To allow for a more fine-grained specification of the preferences, we assume that Σ is

partitioned into priority levels $\Sigma_1, \dots, \Sigma_n$, so that a preference induced by a preference rule from Σ_i is considered more important than one induced by a preference rule from Σ_j with $j > i$, and will thus be preferably kept in the cycle elimination process. Given two facts α and β such that $\alpha \succ_{\Sigma} \beta$, we denote by $level(\alpha, \beta)$ the minimal index i such that $\alpha \succ_{\Sigma_i} \beta$. We consider four cycle resolution techniques:

- **Going up (\succ^u):** Let $\succ^u := \emptyset$ and $i := 1$. Then while $\succ^u \cup \succ_{\Sigma_i}$ is acyclic, let $\succ^u := \succ^u \cup \succ_{\Sigma_i}$ and increment i .
- **Going down (\succ^d):** Let $\succ^d := \succ_{\Sigma}$ and $i := n$. Then while \succ^d is cyclic, let $\succ^d := \succ^d \setminus \{(\alpha, \beta) \mid level(\alpha, \beta) = i, (\alpha, \beta) \text{ is in a cycle w.r.t. } \succ^d\}$ and decrement i .
- **Refined going up (\succ^{ru}):** Let $\succ^{ru} := \succ_{\Sigma_1}$, then remove every (α, β) that occurs in a cycle w.r.t. \succ_{Σ_1} . Then for $i = 2$ to n , add to \succ^{ru} all pairs (α, β) such that $level(\alpha, \beta) = i$ and (α, β) does not belong to any cycle w.r.t. $\succ^{ru} \cup \succ_{\Sigma_i}$.
- **Grounded (\succ^g):** Let $\succ^g := \emptyset$. Then until a fixpoint is reached, add to \succ^g all pairs (α, β) such that $\alpha \succ_{\Sigma} \beta$ and for every cycle c of \succ_{Σ} containing (α, β) , either there is $(\gamma, \delta) \in c$ such that $level(\alpha, \beta) < level(\gamma, \delta)$, or there is $(\gamma, \delta) \in c$ such that $\succ^g \cup \{(\gamma, \delta)\}$ is cyclic.

We relate these cycle removal strategies to notions that have been proposed in the literature to select a single consistent set of facts from a knowledge base whose dataset is partitioned into priority levels [15, 12], and show that $\succ^u \subseteq \succ^d \subseteq \succ^g$ and $\succ^u \subseteq \succ^d \subseteq \succ^{ru}$, and that each of these relations can be computed in polynomial time from the relations \succ_{Σ_i} .

3. ASP Implementation and Experiments

We implement our approach using answer set programming (ASP) [16, 17] to evaluate the preference rules, apply the desired cycle resolution techniques to obtain a priority relation, and answer queries under optimal repair-based semantics (AR, IAR or brave semantics based on Pareto- or completion-optimal repairs), towards an end-to-end system for querying inconsistent knowledge bases. Our system takes as input logic programs representing the input, and computes the query answers under the chosen semantics w.r.t. \succ^x for the chosen $x \in \{u, d, ru, g\}$. All building blocks can be encoded into ASP programs that a Python program combines and passes to the ASP solver clingo [18] to check whether the resulting program has a stable model. However, we found more efficient in practice to split the computation into several steps and implement some of them in Python. Our approach applies to any logical theory \mathcal{T} such that:

1. there exists a set $Inc(\mathcal{T})$ of rules of the form $q \rightarrow \perp$ with q a Boolean CQ, such that for every dataset \mathcal{D} , $(\mathcal{D}, \mathcal{T}) \models \perp$ iff there exists $q \rightarrow \perp \in Inc(\mathcal{T})$ such that $\mathcal{D} \models q$; and
2. for every CQ $q(\vec{x})$ there exists a set $Rew(q, \mathcal{T})$ of rules of the form $q'(\vec{x}) \rightarrow q(\vec{x})$ with q' a CQ such that for every \mathcal{D} s.t. $(\mathcal{D}, \mathcal{T}) \not\models \perp$ and tuple \vec{a} , $(\mathcal{D}, \mathcal{T}) \models q(\vec{a})$ iff there exists $q'(\vec{x}) \rightarrow q(\vec{x}) \in Rew(q, \mathcal{T})$ s.t. $(\mathcal{D}, \mathcal{T}) \models q'(\vec{a})$.

These conditions are fulfilled, e.g., when \mathcal{T} is a set of denial constraints (then, $Inc(\mathcal{T}) = \mathcal{T}$ and $Rew(q, \mathcal{T}) = \{q \rightarrow q\}$), or when \mathcal{T} is a DL-Lite ontology. Regarding preference rules, we handle rules whose bodies are CQs with negation and comparison operators. We expect that the KB $\mathcal{K} = (\mathcal{D}, \mathcal{T})$, meta-database $\mathcal{M} = (\mathbf{id}, \mathcal{F})$, preference rules $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_n$, and query q are all given as ASP programs.

Our main goal is to compare the different approaches to obtaining a priority relation from preferences rules, in terms of run time and size of the priority relation. We also compare our ASP implementation of the optimal repair-based semantics with ORBITS, the existing SAT-based implementation. We use the CQAPri benchmark [19], a synthetic benchmark adapted from LUBM₂₀³ [20] to evaluate inconsistency-tolerant query answering over DL-Lite KBs. We also consider its extension with two priority relations given by the ORBITS benchmark [14] for the comparison with ORBITS, and add a denial constraint to experiment with non-binary conflicts. For the meta-data, we randomly generate facts of the form $date(\mathbf{id}(\alpha), n)$, $source(\mathbf{id}(\alpha), k)$ and $reliability(k, m)$. We define four preference rules which

express that one prefers more recent facts, facts with a more reliable source, $FPr(y)$ over $APr(y)$ facts, and $APr(y)$ over $GrSt(y)$ facts, and partition the rules in one, two or three priority levels.

We were not able to compute \succ^{ru} even on the simplest case because it overflows the number of atoms clingo can handle. However, we managed to compute the other priority relations for almost all small datasets ($>75K$) and several medium size datasets ($>463K$), even in cases with a large proportion of facts in conflict. Interestingly, on all instances for which we computed them, \succ^g never compares more than 5% more pairs of facts than \succ^d , while \succ^u is often reduced to the empty relation. From a computational point of view, \succ^d is significantly faster to compute than \succ^g and \succ^u . This indicates that \succ^d may be a good method to use in practice. Regarding the computation of optimal repair-based semantics, our system is by far slower than ORBITS. However, we note that we manage to answer some queries under AR and brave semantics based on completion-optimal repairs when ORBITS runs out of time or memory.

Acknowledgments

This work was supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014).

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] M. Bienvenu, C. Bourgaux, K. Inoue, R. Jean, A rule-based approach to specifying preferences over conflicting facts and querying inconsistent knowledge bases, in: Proceedings of KR (to appear), 2025.
- [2] L. E. Bertossi, Database repairs and consistent query answering: Origins and further developments, in: Proceedings of PODS, 2019.
- [3] M. Bienvenu, A short survey on inconsistency handling in ontology-mediated query answering, *Künstliche Intelligenz* 34 (2020) 443–451.
- [4] A. Lopatenko, L. E. Bertossi, Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics, in: Proceedings of ICDT, 2007.
- [5] J. Du, G. Qi, Y. Shen, Weight-based consistent query answering over inconsistent SHIQ knowledge bases, *Knowl. Inf. Syst.* 34 (2013) 335–371.
- [6] M. Bienvenu, C. Bourgaux, F. Goasdoué, Querying inconsistent description logic knowledge bases under preferred repair semantics, in: Proceedings of AAI, 2014.
- [7] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, *Artif. Intell.* 312 (2022) 103772.
- [8] T. Lukasiewicz, E. Malizia, C. Molinaro, Complexity of inconsistency-tolerant query answering in datalog \pm under preferred repairs, in: Proceedings of KR, 2023.
- [9] S. Staworko, J. Chomicki, J. Marcinkowski, Prioritized repairing and consistent query answering in relational databases, *Annals of Mathematics and Artificial Intelligence (AMAI)* 64 (2012) 209–246.
- [10] B. Kimelfeld, E. Livshits, L. Peterfreund, Detecting ambiguity in prioritized database repairing, in: Proceedings of ICDT, 2017.
- [11] B. Kimelfeld, E. Livshits, L. Peterfreund, Counting and enumerating preferred database repairs, *Theor. Comput. Sci.* 837 (2020) 115–157.
- [12] M. Bienvenu, C. Bourgaux, Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation, in: Proceedings of KR, 2020.
- [13] M. Bienvenu, C. Bourgaux, Inconsistency handling in prioritized databases with universal constraints: Complexity analysis and links with active integrity constraints, in: Proceedings of KR, 2023.

- [14] M. Bienvenu, C. Bourgaux, Querying inconsistent prioritized data with ORBITS: algorithms, implementation, and experiments, in: Proceedings of KR, 2022.
- [15] S. Benferhat, Z. Bouraoui, K. Tabia, How to select one preferred assertional-based repair from inconsistent and prioritized DL-Lite knowledge bases?, in: Proceedings of IJCAI, 2015.
- [16] V. Lifschitz, Answer Set Programming, Springer, 2019. URL: <https://doi.org/10.1007/978-3-030-24658-7>. doi:10.1007/978-3-030-24658-7.
- [17] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Answer Set Solving in Practice, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2012. URL: <https://doi.org/10.2200/S00457ED1V01Y201211AIM019>. doi:10.2200/S00457ED1V01Y201211AIM019.
- [18] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, M. Schneider, Potassco: The potsdam answer set solving collection, AI Commun. 24 (2011) 107–124.
- [19] C. Bourgaux, Inconsistency Handling in Ontology-Mediated Query Answering. (Gestion des incohérences pour l'accès aux données en présence d'ontologies), Ph.D. thesis, University of Paris-Saclay, France, 2016.
- [20] C. Lutz, I. Seylan, D. Toman, F. Wolter, The combined approach to OBDA: Taming role hierarchies using filters, in: Proceedings of ISWC, 2013.