Towards an End-to-End ASP-Based System for Handling Inconsistent Prioritized Data

Meghyn Bienvenu[1], Camille Bourgaux[2], Katsumi Inoue[3], Robin Jean[1] and Giuseppe Mazzotta[4]

[1]*Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France*
[2]*DI ENS, ENS, CNRS, PSL University & Inria, Paris, France*
[3]*National Institute of Informatics, Tokyo, Japan*
[4]*University of Calabria, Rende, Italy*

Abstract

We present our recent work towards an end-to-end ASP-based system for handling inconsistent data. Our first contribution, presented in [1], introduces a declarative rule-based framework for specifying and computing a priority relation between conflicting facts. Such priority relations have been used to define three kinds of optimal repairs (Pareto-, globally-, and completion-optimal), which can be used in place of classical repairs to define repair-based semantics. Our second contribution, presented in [2], is an implementation of the optimal repair-based variants of three such semantics (AR, brave and IAR) using answer set programming (ASP) and its extension ASP(Q). In particular, this is the first implementation of the globally-optimal repair-based semantics, which are computationally more challenging. We also implement the grounded semantics, a tractable under-approximation of the optimal repair-based semantics rooted in abstract argumentation. These two components are key steps towards a complete pipeline for handling inconsistent data using priority relations. This extended abstract summarizes the main ideas behind each component and the takeaways from our experiments.

Keywords

Inconsistency-tolerant query answering, Optimal repairs, Priority relations, Preference specification, Answer set programming, ASP(Q)

1. Introduction

Repair-based semantics are a prominent means of obtaining meaningful answers to queries which are evaluated over data which is inconsistent w.r.t. some logical theory, both in the relational database and ontology-mediated query answering setting (cf. [3, 4] for brief overviews). In this context, a repair is a subset-maximal subset of the data consistent with the logical theory. The most well-known repair-based semantics, called AR in the KR community and adapted from the consistent query answering approach used in the database area [5], requires that the query holds in every repair [6], while the less cautious brave semantics requires that it holds in some repair [7], and the more cautious IAR semantics that it holds in the intersection of all repairs [6]. Several notions of preferred repairs have been proposed to take into account some preference information and consider only a subset of all the possible repairs to evaluate the queries (cf. [8] for a survey). In particular, Staworko, Chomicki, and Marcinkowski [9] introduced three kinds of optimal repairs based on a priority relation between conflicting facts, which have attracted a lot of interest in the last decade with extensive complexity analyses [10, 11, 12, 13] and a SAT-based implementation [14]. Our work addresses two gaps towards the practical deployment of this framework: (1) users had no simple way to *specify* the priority relation, while it is not realistic to expect users to manually input a binary relation between the facts, and (2) the existing SAT-based implementation [14] is limited to Pareto- and completion-optimal repairs and to the case where the conflicts (i.e., minimal subsets of the data inconsistent with the logical theory) are binary. Indeed, the semantics based on globally-optimal repairs have a higher computational complexity (Σ_2^p/Π_2^p vs. NP/coNP for those based on the other kinds of optimal repairs). This extended abstract summarizes

LINDA 2026: Logical approaches to handling INconsistent DAta Workshop, July 24th, 2026, Lisbon, Portugal
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); giuseppe.mazzotta@unical.it (G. Mazzotta)


two recent papers [1, 2] that together overcome these obstacles towards a coherent end-to-end system, leveraging the well-known answer set programming (ASP) paradigm and its extension with quantifiers, ASP(Q). We describe in Section 3 a rule-based framework for specifying and computing priority relations, and in Section 4 the ASP(Q) encodings used to compute query answers under the optimal-repair based semantics. We additionally provide an ASP implementation of the grounded semantics [12], a tractable under-approximation of the optimal repair-based semantics rooted in abstract argumentation. We also explain in Section 4 how to integrate all our components into a full pipeline. Finally, Section 5 summarizes our main experimental findings, and we discuss future work in Section 6.

2. Preliminaries

Prioritized KBs and optimal repairs. A knowledge base (KB) $\mathcal{K} = \langle \mathcal{D}, \mathcal{T} \rangle$ consists of a dataset $\mathcal{D}$ and a logical theory $\mathcal{T}$ (an ontology or a set of constraints). A *conflict* of $\mathcal{K}$ is a minimal subset $\mathcal{C} \subseteq \mathcal{D}$ such that $\langle \mathcal{C}, \mathcal{T} \rangle$ is inconsistent, and a *repair* is a maximal subset $\mathcal{R} \subseteq \mathcal{D}$ such that $\langle \mathcal{R}, \mathcal{T} \rangle$ is consistent. A *priority relation* $\succ$ for $\mathcal{K}$ is an acyclic binary relation over $\mathcal{D}$ such that $\alpha \succ \beta$ implies $\{\alpha, \beta\}$ is included in some conflict. It is *total* if for every pair $\alpha \neq \beta$ such that $\{\alpha, \beta\}$ is included in a conflict, either $\alpha \succ \beta$ or $\beta \succ \alpha$. A *completion* of $\succ$ is a total priority relation $\succ' \supseteq \succ$. A *prioritized KB* $\mathcal{K}_\succ$ is a KB $\mathcal{K}$ with a priority relation $\succ$ for $\mathcal{K}$. A repair $\mathcal{R}$ admits a *Pareto improvement* $\mathcal{R}'$ if $\langle \mathcal{R}', \mathcal{T} \rangle$ is consistent and there exists $\beta \in \mathcal{R}' \setminus \mathcal{R}$ such that $\beta \succ \alpha$ for every $\alpha \in \mathcal{R} \setminus \mathcal{R}'$. It admits a *global improvement* $\mathcal{R}'$ if $\langle \mathcal{R}', \mathcal{T} \rangle$ is consistent, $\mathcal{R}' \neq \mathcal{R}$, and for every $\alpha \in \mathcal{R} \setminus \mathcal{R}'$, there exists $\beta \in \mathcal{R}' \setminus \mathcal{R}$ with $\beta \succ \alpha$. A repair $\mathcal{R}$ is Pareto-optimal (resp. globally-optimal) if it does not admit a Pareto (resp. global) improvement. It is completion-optimal if it is a globally-optimal repair of $\mathcal{K}_{\succ'}$, for some completion $\succ'$ of $\succ$. We denote by $SRep(\mathcal{K})$, $PRep(\mathcal{K}_\succ)$, $GRep(\mathcal{K}_\succ)$ and $CRep(\mathcal{K}_\succ)$ the sets of (standard) repairs and Pareto-, globally- and completion-optimal repairs, respectively. It is known that $CRep(\mathcal{K}_\succ) \subseteq GRep(\mathcal{K}_\succ) \subseteq PRep(\mathcal{K}_\succ) \subseteq SRep(\mathcal{K})$.

X-brave, X-AR and X-IAR semantics. Fix $X \in \{S, P, G, C\}$ and consider a prioritized KB $\mathcal{K}_\succ$ with $\mathcal{K} = \langle \mathcal{D}, \mathcal{T} \rangle$, query $q(\vec{x})$, and tuple of constants $\vec{a}$. Then $\vec{a}$ is an answer to $q(\vec{x})$ over $\mathcal{K}_\succ$ under *X-brave semantics* if $\langle \mathcal{R}, \mathcal{T} \rangle \models q(\vec{a})$ for some $\mathcal{R} \in XRep(\mathcal{K}_\succ)$; under *X-AR semantics*, if $\langle \mathcal{R}, \mathcal{T} \rangle \models q(\vec{a})$ for every $\mathcal{R} \in XRep(\mathcal{K}_\succ)$; under *X-IAR semantics*, if $\langle \mathcal{B}, \mathcal{T} \rangle \models q(\vec{a})$ where $\mathcal{B} = \bigcap_{\mathcal{R} \in XRep(\mathcal{K}_\succ)} \mathcal{R}$. In FOL fragments for which KB consistency checking and query entailment are in PTime w.r.t. data complexity (e.g., DL-Lite or denial constraints), the data complexity of query entailment under these semantics is in NP (X-brave) and coNP (X-AR/X-IAR) for $X \in \{P, C\}$, but in Σ_2^p (G-brave) and Π_2^p (G-AR/G-IAR) when globally-optimal repairs are used [9, 12].

Grounded semantics. The grounded semantics for prioritized KBs [12] was defined via translation to abstract argumentation frameworks. For a prioritized KB $\mathcal{K}_\succ$ with $\mathcal{K} = \langle \mathcal{D}, \mathcal{T} \rangle$, the *attack relation* $\rightsquigarrow \subseteq (2^{\mathcal{D}} \setminus \{\emptyset\}) \times \mathcal{D}$ is defined by: $\mathcal{B} \rightsquigarrow \alpha$ iff $\mathcal{B} \cup \{\alpha\}$ is a conflict and for every $\beta \in \mathcal{B}$, $\alpha \not\succ \beta$. The characteristic function $\Gamma : 2^{\mathcal{D}} \mapsto 2^{\mathcal{D}}$ is defined by $\Gamma(\mathcal{B}) = \{\alpha \mid \mathcal{E} \rightsquigarrow \alpha \Rightarrow \exists \mathcal{F} \subseteq \mathcal{B}, \beta \in \mathcal{E} \text{ s.t. } \mathcal{F} \rightsquigarrow \beta\}$. The *grounded repair* of $\mathcal{K}_\succ$ is the inclusion-minimal $\mathcal{G} \subseteq \mathcal{D}$ such that $\langle \mathcal{G}, \mathcal{T} \rangle$ is consistent and $\mathcal{G} = \Gamma(\mathcal{G})$, or equivalently, the least fixpoint of Γ. A tuple $\vec{a}$ is an answer to $q(\vec{x})$ over $\mathcal{K}_\succ$ under *grounded semantics* if $\langle \mathcal{G}, \mathcal{T} \rangle \models q(\vec{a})$. It is known that answers that hold under the grounded semantics hold under X-IAR for $X \in \{P, G, C\}$ and that in FOL fragments for which the size of conflicts is bounded independently from the data, and for which KB consistency and query entailment are in PTime w.r.t. data complexity, query entailment under grounded semantics is in PTime w.r.t. data complexity [12].

3. Specifying Priority Relations via Preference Rules

Our first contribution addresses the question: *how can a user specify a priority relation?* We propose a declarative *preference rule* framework, where rules of the form $\mathsf{Cond}(x_1, x_2) \rightarrow \mathsf{pref}(x_1, x_2)$ express that, when certain conditions $\mathsf{Cond}(x_1, x_2)$ hold, the fact with identifier x_1 is preferred over the fact with identifier x_2. The body $\mathsf{Cond}(x_1, x_2)$ of a preference rule may reference the dataset itself (e.g., checking

the presence or absence of some facts), as well as a *meta-database* providing auxiliary information about the facts (such as timestamps or sources), and the axioms entailed by the ontology.

Example 1. *Consider a KB $\mathcal{K} = \langle \mathcal{D}, \mathcal{T} \rangle$ whose ontology expresses that associate and full professors (*APr *and* FPr*) are faculty members (*Fac*), clerical staff workers (*Cleric*) are administrative staff workers (*Adm*), and one cannot be both an associate and a full professor, or an administrative staff worker and a faculty member. Assume that we have access to a meta-database $\mathcal{M} = (\mathbf{id}, \mathcal{F})$, where $\mathbf{id}$ is a function that associates an identifier to each fact of $\mathcal{D}$ and $\mathcal{F}$ records the year that facts have been added to the university database using the predicate* Date*. We could define the following preference rules.*

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

The first rule, which uses the binary predicate $<$ to compare dates, states a general preference for keeping more recently added facts. The second one states if we have both $\mathsf{FPr}(p)$ and $\mathsf{APr}(p)$, we prefer to keep $\mathsf{FPr}(p)$, 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*).*

Our preference rules are syntactically very similar to those define in [15]. However, the latter are evaluated over the repairs themselves, to directly define a preference relation between repairs, whereas our rules are evaluated over the dataset (and meta-database) and yield a priority relation between facts, which is then lifted to get optimal repairs (see [8, Section 5.3] for more discussion).

Given a set of preference rules Σ, we denote by $\succ_\Sigma$ the binary relation over facts obtained from the preferences over facts induced by Σ, restricted to facts that appear together in some conflict. This relation may still fail to be a priority relation if it contains a cycle. 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 $\succ_\Sigma$.

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 $\succ_\Sigma$ 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 $\succ_\Sigma$, 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. We 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 $\Sigma_1, \ldots, \Sigma_n$, 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. We consider four cycle resolution techniques that we call (in reference to the different ways they use the priority levels to remove cycles) *going up* ($\succ^u$), *going down* ($\succ^d$), *refined going up* ($\succ^{ru}$), and *grounded* ($\succ^g$). We relate them 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 [16, 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 $\succ_{\Sigma_i}$.

4. Implementation in ASP and ASP(Q)

Our second contribution is an implementation of the key components of the full framework using answer set programming (ASP), a declarative programming paradigm in which problems are encoded as logic programs whose *answer sets* (stable models) represent solutions [17, 18], and its extension with quantifiers ASP(Q) [19]. ASP has already been successfully applied to consistent query answering [20, 21, 22] using standard repairs. However, it requires notoriously complex encoding techniques such as saturation [23, 24] to handle problems in the second level of the polynomial hierarchy, while ASP(Q) provides a natural and compact alternative for encoding such problems, making it a good candidate for implementing semantics based on globally-optimal repairs.

Priority relation computation. We used ASP to implement the evaluation of preference rules and cycle resolution techniques from Section 3. Our approach applies to any logical theory $\mathcal{T}$ such that there exists a set $Inc(\mathcal{T})$ of rules $q \rightarrow \bot$ with q a Boolean CQ (possibly with inequality atoms), such that for every dataset $\mathcal{D}$, $\langle \mathcal{D}, \mathcal{T} \rangle \models \bot$ iff $\mathcal{D} \models q$ for some $q \rightarrow \bot \in Inc(\mathcal{T})$ (e.g., when $\mathcal{T}$ is a set of denial constraints, in which case $Inc(\mathcal{T}) = \mathcal{T}$, or a DL-Lite ontology). Regarding preference rules, we handle rules whose bodies are CQs with negation and comparison operators. The system takes as input ASP programs encoding the dataset, constraints $Inc(\mathcal{T})$, meta-database, and preference rules, and outputs the priority relation obtained by the chosen cycle resolution strategy ($\succ^u$, $\succ^d$, $\succ^{ru}$, or $\succ^g$).

X-AR and X-brave semantics. We define ASP and ASP(Q) encodings to compute the answers that hold under the considered optimal repair-based semantics from their causes (i.e., minimal subsets $\mathcal{C} \subseteq \mathcal{D}$ such that $\langle \mathcal{C}, \mathcal{T} \rangle$ is consistent and entails $q(\vec{a})$), the conflicts and the priority relation. For $X \in \{P, C\}$, query answering is in (co)NP and we use plain ASP programs built from modular building blocks which (1) localize the computation to a set of relevant, *reachable*, facts $\mathcal{F}$ (defined via a reachability notion already used in [14] from the causes, conflicts, and priority relation), (2) guess a subset $\mathcal{R} \subseteq \mathcal{F}$ that does not contain any conflict, (3) enforce that $\mathcal{R}$ is Pareto-optimal (resp. completion-optimal) in $\mathcal{F}$, and (4) contains some cause (for X-brave semantics) or does not contain any cause (for X-AR semantics). For globally-optimal repairs, we use an ASP(Q) encoding. In this case, we needed to use another reachability notion to localize the problems. Our encodings are of the following form.

$$\Pi^G_{brave} = \exists^{st}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{SomeCause})\ \forall^{st}\Pi_{GImp} : C$$

$$\Pi^G_{AR} = \exists^{st}(\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{NoCause})\ \forall^{st}\Pi_{GImp} : C$$

Here $\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{SomeCause}$ (resp. $\Pi_{Reach} \cup \Pi_{Rep} \cup \Pi_{NoCause}$) guesses a repair $\mathcal{R}$ of the set of reachable facts that contains some cause (resp. does not contain any cause), Π_{GImp} guesses a global improvement of $\mathcal{R}$, and C is a constraint that is never satisfied by answer sets of the programs. Intuitively, since C is always incoherent, Π^G_{brave} is coherent iff the universal quantification is trivially satisfied, i.e., $\mathcal{R}$ has no global improvement, and similarly for Π^G_{AR}.

X-IAR. We compute $\bigcap_{\mathcal{R} \in XRep(\mathcal{K}_\succ)} \mathcal{R}$ by checking for each fact whether it holds under X-AR (since a fact holds under X-AR iff it is in every optimal repair). It is then possible to use this set to evaluate the queries under the X-IAR semantics.

Tractable under-approximations of P-IAR. We use a preprocessing step that identifies a subset of the answers that hold under P-IAR (hence under all the considered semantics). We consider two tractable such under-approximations: the *grounded semantics*, recalled in Section 2, and the *trivially P-IAR answers* [25, 14], which have some cause such that none of its fact is attacked (w.r.t. $\rightsquigarrow$, i.e., is in $\Gamma(\emptyset)$). In practice, we compute the grounded repair $\mathcal{G}$ or the set of trivially P-IAR facts $\Gamma(\emptyset)$, then use it to filter the answers that have some cause included in it.

Pipeline. The components described in this section form the key ingredients of a pipeline for optimal repair-based inconsistency-tolerant query answering, which takes as input a dataset $\mathcal{D}$, a set of constraints $Inc(\mathcal{T})$, a meta-database $\mathcal{M}$, a set of preference rules $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_n$, and a query q. All inputs are given as ASP programs, and a Python orchestrator combines and passes the relevant components to the ASP or ASP(Q) solver depending on the step and the chosen semantics. The whole

pipeline is as follows. A preprocessing step computes the conflicts of $\mathcal{K}$, evaluates the preference rules and applies the chosen cycle resolution strategy to obtain the priority relation $\succ$. Then, either the grounded repair $\mathcal{G}$ (or the set of trivially P-IAR facts $\Gamma(\emptyset)$) is computed. When a query arrives, its potential answers and causes are computed and $\mathcal{G}$ (or $\Gamma(\emptyset)$) is used to check for each potential answer whether it is grounded (or trivially P-IAR), to avoid the use of ASP/ASP(Q) encodings for as many answers as possible. For the remaining potential answers, the appropriate ASP or ASP(Q) encoding is invoked to determine if it holds under the chosen semantics.

5. Experimental Evaluation

We evaluated our approach using the CQAPri benchmark [26], a synthetic benchmark adapted from $\text{LUBM}^{\exists}_{20}$ [27] for inconsistency-tolerant query answering over DL-Lite KBs, which was extended with priority relations [14]. To experiment also with non-binary conflicts, we added a denial constraint that yields conflicts of size 10. As ASP(Q) solver we used CASPER [28], and as ASP solver, CLINGO [29].

Priority relation computation. On small datasets (75K facts) and medium-size datasets (463K facts), we were able to compute $\succ^u$, $\succ^d$, and $\succ^g$ in most cases, even in cases with more than 29% of facts in conflicts. However, we were not able to compute $\succ^{ru}$ even on the simplest cases as it overflows the atom count of CLINGO. Interestingly, $\succ^d$ and $\succ^g$ often coincide and never differ by more than 5% of compared facts on instances for which we computed them, 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 $\succ^u$, suggesting it may be a good method in practice.

Query answering. A key finding is the surprising effectiveness of the grounded semantics as an approximation: a large proportion of potential answers hold under grounded semantics, and the grounded repair can be computed efficiently in most cases. Moreover, even in the cases where we were not able to compute the grounded repair, we were able to compute the trivially P-IAR facts very efficiently, and the proportion of facts that belong to the grounded repair but are not trivially P-IAR varies from 1%-2% on the datasets with few conflicts to 15%-31% on those with the highest proportion of facts in conflicts. Finally, the grounded repair covers a large fraction of the intersection of optimal repairs. Given the cost of checking whether each non-grounded fact holds under X-(I)AR, this suggests the interest of using grounded rather than X-IAR. Another observation is that using globally-optimal repairs is significantly more costly than using Pareto- or completion-optimal repairs (e.g., deciding whether a query holds under G-AR took more than 200s in 51% of the cases while it took at most 14s for P-AR), and the gap grows with the proportion of conflicting facts. This suggests that even if we aim to compute G-AR and G-brave answers, a better strategy is to first test whether the candidate answer holds under P-AR and P-brave (which are the fastest optimal repair semantics and since answers that hold under P-AR also hold under G-AR while answers that do not hold under P-brave do not hold under G-brave) and only call the G-AR and G-brave procedures if the status remains unresolved. Finally, unsurprisingly, localization using reachable facts is crucial for feasibility even on the smallest datasets.

6. Perspectives

There are several directions for future work. From a theoretical point of view, we could extend the static analysis of preference rules, by considering more classes of logical theories and preference rules. Besides the problem of deciding whether a theory and set of preference rules ensure that the induced relation is acyclic, one could wonder whether they guarantee that there exists a unique optimal repair. From a practical point of view, given the interest of the grounded repair, which makes it possible to employ a principled inconsistency-tolerant semantics with little overhead compared to standard query answering, it would be interesting to develop highly optimized methods for computing and updating the grounded repair for very large datasets. Finally, there remain a few missing pieces to have a truly end-to-end system for query answering over inconsistent KBs with preference rules, in particular to generate the input logic programs from data and logical theory given in various formats.

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, 2025. Extended version with appendix available at: https://arxiv.org/abs/2508.07742.

[2] M. Bienvenu, C. Bourgaux, R. Jean, G. Mazzotta, Using ASP(Q) to handle inconsistent prioritized data, in: Proceedings of KR, 2026. Extended version with appendix available at: https://arxiv.org/abs/2604.21603.

[3] L. E. Bertossi, Database repairs and consistent query answering: Origins and further developments, in: Proceedings of PODS, 2019.

[4] M. Bienvenu, A short survey on inconsistency handling in ontology-mediated query answering, Künstliche Intelligenz 34 (2020) 443–451.

[5] M. Arenas, L. E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proceedings of PODS, 1999.

[6] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. F. Savo, Inconsistency-tolerant semantics for description logics, in: Proceedings of RR, 2010.

[7] M. Bienvenu, R. Rosati, Tractable approximations of consistent query answering for robust ontology-based data access, in: Proceedings of IJCAI, 2013.

[8] C. Bourgaux, Inconsistency-tolerant semantics based on (preferred) repairs, in: Proceedings of RW, 2025.

[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] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, Artif. Intell. 312 (2022) 103772.

[16] 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.

[17] 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.

[18] 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.

[19] G. Amendola, F. Ricca, M. Truszczynski, Beyond NP: quantifying over answer sets, Theory Pract. Log. Program. 19 (2019) 705–721.

[20] T. Eiter, M. Fink, G. Greco, D. Lembo, Repair localization for query answering from inconsistent databases, ACM Trans. Database Syst. 33 (2008) 10:1–10:51.

[21] M. C. Marileo, L. E. Bertossi, The consistency extractor system: Answer set programs for consistent query answering in databases, Data Knowl. Eng. 69 (2010) 545–572.

[22] M. Manna, F. Ricca, G. Terracina, Consistent query answering via ASP from different perspectives: Theory and practice, Theory Pract. Log. Program. 13 (2013) 227–252.

[23] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Ann. Math. Artif. Intell. 15 (1995) 289–323.

[24] M. Gebser, R. Kaminski, T. Schaub, Complex optimization in answer set programming, Theory Pract. Log. Program. 11 (2011) 821–839.
[25] M. Bienvenu, C. Bourgaux, F. Goasdoué, Querying inconsistent description logic knowledge bases under preferred repair semantics, in: Proceedings of AAAI, 2014.
[26] 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.
[27] C. Lutz, I. Seylan, D. Toman, F. Wolter, The combined approach to OBDA: Taming role hierarchies using filters, in: Proceedings of ISWC, 2013.
[28] C. Andrea, G. Mazzotta, F. Ricca, 2-ASP(Q) solving based on CEGAR, in: Proceedings of AAAI, 2026.
[29] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, CoRR abs/1705.09811 (2017).