Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Using ASP(Q) to Handle Inconsistent Prioritized Data

Meghyn Bienvenu[1] , **Camille Bourgaux**[2] , **Robin Jean** [1] , **Giuseppe Mazzotta**[3]
[1]Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France
[2]DI ENS, ENS, CNRS, PSL University & Inria, Paris, France
[3]University of Calabria, Rende, Italy
{meghyn.bienvenu, robin.jean}@u-bordeaux.fr, camille.bourgaux@ens.fr, giuseppe.mazzotta@unical.it

Abstract

We explore the use of answer set programming (ASP) and its extension with quantifiers, ASP(Q), for inconsistency-tolerant querying of prioritized data, where a priority relation between conflicting facts is exploited to define three notions of optimal repairs (Pareto-, globally- and completion-optimal). We consider the variants of three well-known semantics (AR, brave and IAR) that use these optimal repairs, and for which query answering is in the first or second level of the polynomial hierarchy for a large class of logical theories. Notably, this paper presents the first implementation of globally-optimal repair-based semantics, as well as the first implementation of the grounded semantics, which is a tractable under-approximation of all these optimal repair-based semantics. Our experimental evaluation sheds light on the feasibility of computing answers under globally-optimal repair semantics and the impact of adopting different semantics, approximations, and encodings.

1 Introduction

Repair-based semantics are a prominent means of obtaining meaningful answers to queries posed over some data which is inconsistent w.r.t. some logical theory, both in the relational database and ontology-mediated query answering setting (cf. (Bertossi 2019; Bienvenu 2020) 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, requires that the query holds in every repair, while the less cautious brave semantics requires that it holds in some repair, and the more cautious IAR semantics that it holds in the intersection of all repairs (Arenas, Bertossi, and Chomicki 1999; Lembo et al. 2010; Bienvenu and Rosati 2013). 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. (Bourgaux 2025) for a survey). In particular, Staworko, Chomicki, and Marcinkowski (2012) 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 (Kimelfeld, Livshits, and Peterfreund 2017; Kimelfeld, Livshits, and Peterfreund 2020; Bienvenu and Bourgaux 2020; Bienvenu and Bourgaux 2023), two implementations (Bienvenu and Bourgaux 2022; Bienvenu et al. 2025), and a framework for specifying and computing priority relations (Bienvenu et al. 2025).

However, only two of these three kinds of repairs have actually been implemented in the existing systems, namely the Pareto- and completion-optimal repairs. This is due to the higher complexity of reasoning with the third kind of repairs, called globally-optimal. Indeed, for a large class of database constraints and ontology languages, the data complexity of query answering under the variants of AR, IAR and brave that use Pareto- or completion-optimal repairs is in the first level of the polynomial hierarchy, while it is in the second level for the globally-optimal repair-based variants.

Most implementations of repair-based semantics encode query entailment under some intractable semantics into SAT, binary integer programming, or answer set programming (ASP) to take advantage of the efficient solvers that exist for these problems (see, e.g., (Greco, Greco, and Zumpano 2003; Eiter et al. 2008; Manna, Ricca, and Terracina 2013; Kolaitis, Pema, and Tan 2013; Bienvenu, Bourgaux, and Goasdoué 2014; Dixit and Kolaitis 2019) for relevant examples and (Bourgaux 2025, Table 8) for an overview). The two implementations of optimal repair-based semantics follow this path: the one by Bienvenu and Bourgaux (2022) is based on the use of tractable approximations and SAT solvers, and that by Bienvenu et al. (2025) uses ASP. A notable difference between the two systems is that the former focuses on the case where the conflicts (i.e., the minimal sets of facts inconsistent w.r.t. the logical theory) are of size at most two while the latter handles conflicts of arbitrary size.

ASP (with disjunctive rules) can be used to model problems up to the second level of the polynomial hierarchy, meaning that semantics based on globally-optimal repairs could theoretically be implemented in ASP. However, this requires advanced techniques, such as saturation (Eiter and Gottlob 1995), which complicate modeling *"beyond the capabilities of the common ASP laymen"* (Gebser, Kaminski, and Schaub 2011). Moreover, efficiency-wise it has been shown that in many cases of practical interest alternative formalisms are preferred to disjunctive ASP (Amendola et al. 2022). Among alternative formalisms, answer set programming with quantifiers (ASP(Q)) uses quantifiers over answer sets of ASP programs and provides a natural and compact way of modeling problems in the entire polynomial hierar-

chy (Amendola, Ricca, and Truszczynski 2019). ASP(Q) has already been used to tackle problems related to planning (Fandinno et al. 2021; Faber, Morak, and Chrpa 2022) and abstract argumentation (Faber 2024).

We investigate the use of ASP(Q) and ASP to implement optimal repair-based semantics. Compared to the previous ASP-based implementation by Bienvenu et al. (2025), we handle globally-optimal repairs thanks to ASP(Q) and improve the system performance by using tractable approximations in a similar way as Bienvenu and Bourgaux (2022). Moreover, we explore the use of another tractable approximation of the optimal repair-based semantics, called the grounded semantics (Bienvenu and Bourgaux 2020), which has never been implemented so far. Our experimental evaluation shows that using globally-optimal repairs is considerably more challenging than for Pareto- or completion-optimal repairs, while at the same time demonstrating the utility of different optimizations, in particular, the surprising effectiveness of the grounded semantics.

Proofs and additional details on the experiments are provided in (Bienvenu et al. 2026). All materials to reproduce the experiments are available from https://github.com/rjean007/InconsistentPrioritizedData-ASPQ.

2 Preliminaries

In this section, we introduce the relevant background on inconsistency-tolerant semantics and ASP(Q).

2.1 Optimal Repair-Based Semantics

We recall the framework of inconsistency-tolerant querying of prioritized knowledge bases. All notions will be illustrated in Example 1. Let $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{V}$ be three disjoint sets of predicates, constants and variables, respectively. We assume that each predicate has some arity $n \geq 1$, and let $\mathbf{P}_n$ be the set of the n-ary predicates in $\mathbf{P}$.

Knowledge Bases, Conflicts, Repairs A *knowledge base (KB)* $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ consists of a *dataset* $\mathcal{D}$ and a *logical theory* $\mathcal{T}$: $\mathcal{D}$ is a finite set of *facts* of the form $P(c_1, \ldots, c_n)$ with $P \in \mathbf{P}_n$, $c_i \in \mathbf{C}$ for $1 \leq i \leq n$, and $\mathcal{T}$ is a finite set of first-order logic (FOL) sentences built from $\mathbf{P}$, $\mathbf{C}$ and $\mathbf{V}$. Typically, $\mathcal{T}$ will be either an *ontology* (e.g., formulated in some description logic) or a set of *database constraints*. In particular, we consider description logics of the *DL-Lite family* (Calvanese et al. 2007) and *denial constraints* of the form $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \bot$, where each α_i is a relational or inequality atom, which include *functional dependencies (FDs)*. A KB $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ is *consistent*, and $\mathcal{D}$ is called $\mathcal{T}$*-consistent*, if $\mathcal{D} \cup \mathcal{T}$ has some model. Otherwise, $\mathcal{K}$ is *inconsistent*, denoted $\mathcal{K} \models \bot$. A *conflict* of $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ is an inclusion-minimal subset $\mathcal{C} \subseteq \mathcal{D}$ such that $(\mathcal{C}, \mathcal{T}) \models \bot$. The set of conflicts of $\mathcal{K}$ is denoted $\mathit{Conf}(\mathcal{K})$. A *(subset) repair* of $\mathcal{K}$ is an inclusion-maximal subset $\mathcal{R} \subseteq \mathcal{D}$ such that $(\mathcal{R}, \mathcal{T}) \not\models \bot$. The set of repairs of $\mathcal{K}$ is denoted $\mathit{SRep}(\mathcal{K})$.

Prioritized KBs, Optimal Repairs A *priority relation* $\succ$ for a KB $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ is an acyclic binary relation over the facts of $\mathcal{D}$ such that $\alpha \succ \beta$ implies that $\{\alpha, \beta\} \subseteq \mathcal{C}$ for some $\mathcal{C} \in \mathit{Conf}(\mathcal{K})$. It is *total* if for every pair $\alpha \neq \beta$ such that $\{\alpha, \beta\} \subseteq \mathcal{C}$ for some $\mathcal{C} \in \mathit{Conf}(\mathcal{K})$, 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 $\mathcal{K}$. Three kinds of optimal repairs are defined:

Definition 1. *Let $\mathcal{K}_\succ$ be a prioritized KB with $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ and $\mathcal{R} \in \mathit{SRep}(\mathcal{K})$.*

- *A* Pareto improvement *of $\mathcal{R}$ is a $\mathcal{T}$-consistent $\mathcal{B} \subseteq \mathcal{D}$ such that there is $\beta \in \mathcal{B} \setminus \mathcal{R}$ with $\beta \succ \alpha$ for every $\alpha \in \mathcal{R} \setminus \mathcal{B}$.*
- *A* global improvement *of $\mathcal{R}$ is a $\mathcal{T}$-consistent $\mathcal{B} \subseteq \mathcal{D}$ such that $\mathcal{B} \neq \mathcal{R}$ and for every $\alpha \in \mathcal{R} \setminus \mathcal{B}$ there exists $\beta \in \mathcal{B} \setminus \mathcal{R}$ such that $\beta \succ \alpha$.*

The repair $\mathcal{R}$ is:

- Pareto-optimal *if there is no Pareto improvement of $\mathcal{R}$;*
- globally-optimal *if there is no global improvement of $\mathcal{R}$;*
- completion-optimal *if $\mathcal{R}$ is a globally-optimal repair of $\mathcal{K}_{\succ'}$, for some completion $\succ'$ of $\succ$.*

We denote by $\mathit{PRep}(\mathcal{K}_\succ)$, $\mathit{GRep}(\mathcal{K}_\succ)$ and $\mathit{CRep}(\mathcal{K}_\succ)$ the sets of Pareto-, globally- and completion-optimal repairs.

It is known that $\mathit{CRep}(\mathcal{K}_\succ) \subseteq \mathit{GRep}(\mathcal{K}_\succ) \subseteq \mathit{PRep}(\mathcal{K}_\succ)$.

Queries, Repair-Based Semantics A *conjunctive query* (CQ) is a conjunction of atoms $P(t_1, \ldots, t_n)$ ($P \in \mathbf{P}_n$, $t_i \in \mathbf{C} \cup \mathbf{V}$), where some variables may be existentially quantified. Given a query $q(\vec{x})$, with free variables $\vec{x}$, and a tuple of constants $\vec{a}$ such that $|\vec{a}| = |\vec{x}|$, $q(\vec{a})$ denotes the first-order sentence obtained by replacing each variable in $\vec{x}$ by the corresponding constant in $\vec{a}$. A *(certain) answer* to $q(\vec{x})$ over $\mathcal{K}$ is a tuple $\vec{a}$ of constants such that $q(\vec{a})$ holds in every model of $\mathcal{K}$, denoted $\mathcal{K} \models q(\vec{a})$. When the KB is inconsistent, we consider the following alternative semantics, parameterized by the considered type of repair.

Definition 2. *Fix $X \in \{S, P, G, C\}$ and consider a prioritized KB $\mathcal{K}_\succ$ with $\mathcal{K} = (\mathcal{D}, \mathcal{T})$, 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*, denoted $\mathcal{K}_\succ \models^X_{\text{brave}} q(\vec{a})$, if $(\mathcal{R}, \mathcal{T}) \models q(\vec{a})$ for some $\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)$;*
- *under* X-AR semantics*, denoted $\mathcal{K}_\succ \models^X_{\text{AR}} q(\vec{a})$, if $(\mathcal{R}, \mathcal{T}) \models q(\vec{a})$ for every $\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)$;*
- *under* X-IAR semantics*, denoted $\mathcal{K}_\succ \models^X_{\text{IAR}} q(\vec{a})$, if $(\mathcal{B}, \mathcal{T}) \models q(\vec{a})$ where $\mathcal{B} = \bigcap_{\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)} \mathcal{R}$.*

It is known that $\mathcal{K}_\succ \models^X_{\text{IAR}} q \Rightarrow \mathcal{K}_\succ \models^X_{\text{AR}} q \Rightarrow \mathcal{K}_\succ \models^X_{\text{brave}} q$. A *cause* for $q(\vec{a})$ w.r.t. $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ is an inclusion-minimal $\mathcal{T}$-consistent subset $\mathcal{C} \subseteq \mathcal{D}$ such that $(\mathcal{C}, \mathcal{T}) \models q(\vec{a})$. The set of causes for $q(\vec{a})$ w.r.t. $\mathcal{K}$ is denoted by $\mathit{Causes}(q(\vec{a}), \mathcal{K})$. We will use the following characterizations of the semantics:

- $\mathcal{K}_\succ \models^X_{\text{brave}} q(\vec{a})$ iff there exist $\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)$ and $\mathcal{C} \in \mathit{Causes}(q(\vec{a}), \mathcal{K})$ such that $\mathcal{C} \subseteq \mathcal{R}$;
- $\mathcal{K}_\succ \not\models^X_{\text{AR}} q(\vec{a})$ iff there exists $\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)$ such that for every $\mathcal{C} \in \mathit{Causes}(q(\vec{a}), \mathcal{K})$, $\mathcal{C} \not\subseteq \mathcal{R}$;
- $\mathcal{K}_\succ \models^X_{\text{IAR}} q(\vec{a})$ iff there exists $\mathcal{C} \in \mathit{Causes}(q(\vec{a}), \mathcal{K})$ such that $\mathcal{C} \subseteq \bigcap_{\mathcal{R} \in \mathit{XRep}(\mathcal{K}_\succ)} \mathcal{R}$.

Theorems 1 and 2 summarize known results on the *data complexity* (where the sizes of the logical theory $\mathcal{T}$ and query $q(\vec{x})$ are assumed to be fixed) of query answering under these semantics (cf. survey (Bourgaux 2025, Table 6)).

Theorem 1. *Let $\mathcal{L}$ be an FOL fragment for which KB consistency and query entailment are in* PTIME*. Query entailment for $\mathcal{L}$ KBs is in Σ_2^p under G-brave semantics, in Π_2^p under G-AR and G-IAR semantics, and for $X \in \{S, P, C\}$, it is in* NP *under X-brave semantics, and in co*NP *under X-AR and X-IAR semantics.*

Theorem 2. *Let $\mathcal{L}$ be any FOL fragment that extends DL-Lite$_{\mathsf{core}}$ or FDs. Query entailment for $\mathcal{L}$ KBs is Σ_2^p-hard under G-brave semantics, Π_2^p-hard under G-AR and G-IAR semantics,* NP*-hard under X-brave semantics for $X \in \{P, C\}$, and co*NP*-hard under X-AR semantics for $X \in \{S, P, C\}$ and under X-IAR semantics for $X \in \{P, C\}$.*

Attack Relation, Grounded Semantics The grounded semantics for prioritized KBs comes from the area of abstract argumentation. For prioritized KB $\mathcal{K}_\succ$ with $\mathcal{K} = (\mathcal{D}, \mathcal{T})$, the *attack relation* $\rightsquigarrow \subseteq (2^{\mathcal{D}} \setminus \{\emptyset\}) \times \mathcal{D}$ is defined by:

$$\rightsquigarrow = \{(\mathcal{C} \setminus \{\alpha\}, \alpha) \mid \mathcal{C} \in \mathit{Conf}(\mathcal{K}), \alpha \in \mathcal{C}, \forall \beta \in \mathcal{C}, \alpha \not\succ \beta\}.$$

We write $\mathcal{B} \rightsquigarrow \alpha$ for $(\mathcal{B}, \alpha) \in \rightsquigarrow$. 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 $\mathcal{G}$ is $\mathcal{T}$-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*, denoted $\mathcal{K}_\succ \models_{\text{GR}} q(\vec{a})$, if $(\mathcal{G}, \mathcal{T}) \models q(\vec{a})$. It is known that $\mathcal{K}_\succ \models_{\text{GR}} q(\vec{a})$ implies $\mathcal{K}_\succ \models^P_{\text{IAR}} q(\vec{a})$, so answers that hold under the grounded semantics hold under X-IAR for $X \in \{P, G, C\}$. Moreover, if $\mathcal{L}$ is an FOL fragment for which the conflicts size is bounded independently from the data and KB consistency and query entailment are in PTIME, then grounded query entailment for $\mathcal{L}$ KBs is in PTIME (Bienvenu and Bourgaux 2020).

Example 1. *Consider the KB $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ with:*

$$\mathcal{D} = \{A(a), B(a), C(a), D(a), A(b), B(b), C(b)\}$$
$$\mathcal{T} = \{A(x) \wedge B(x) \rightarrow \bot,\ B(x) \wedge C(x) \rightarrow \bot,$$
$$C(x) \wedge D(x) \rightarrow \bot,\ D(x) \wedge A(x) \rightarrow \bot\}$$

The conflicts and repairs of $\mathcal{K}$ are as follows:

$$\mathit{Conf}(\mathcal{K}) = \{\{A(a), B(a)\}, \{B(a), C(a)\}, \{C(a), D(a)\}, \{D(a), A(a)\}, \{A(b), B(b)\}, \{B(b), C(b)\}\}$$
$$\mathit{SRep}(\mathcal{K}) = \{\{A(a), C(a), B(b)\}, \{B(a), D(a), B(b)\} \{A(a), C(a), A(b), C(b)\}, \{B(a), D(a), A(b), C(b)\}\}$$

Let us define a priority relation $\succ$ for $\mathcal{K}$ by $A(a) \succ B(a)$, $C(a) \succ D(a)$, $A(b) \succ B(b)$, and $B(b) \succ C(b)$.

$$\mathit{CRep}(\mathcal{K}_\succ) = \mathit{GRep}(\mathcal{K}_\succ) = \{\{A(a), C(a), A(b), C(b)\}\}$$
$$\mathit{PRep}(\mathcal{K}_\succ) = \mathit{GRep}(\mathcal{K}_\succ) \cup \{\{B(a), D(a), A(b), C(b)\}\}$$

We thus obtain, e.g., $\mathcal{K}_\succ \models^G_{\text{AR}} A(a)$ while $\mathcal{K}_\succ \not\models^P_{\text{AR}} A(a)$, $\mathcal{K}_\succ \models^P_{\text{brave}} B(a)$ while $\mathcal{K}_\succ \not\models^G_{\text{brave}} B(a)$, and $\mathcal{K}_\succ \models^X_{\text{IAR}} C(b)$ for $X \in \{P, G, C\}$. The grounded repair of $\mathcal{K}_\succ$ is $\mathcal{G} = \{A(b), C(b)\}$. Indeed, the attack relation $\rightsquigarrow$ is as written below, so $\Gamma(\emptyset) = \{A(b)\}$, since $A(b)$ is the only fact of $\mathcal{D}$ that is not attacked, $\Gamma(\{A(b)\}) = \{A(b), C(b)\}$, and $\Gamma(\{A(b), C(b)\}) = \{A(b), C(b)\}$.

$$\{A(a)\} \rightsquigarrow B(a) \quad \{B(a)\} \rightsquigarrow C(a) \quad \{C(a)\} \rightsquigarrow B(a)$$
$$\{C(a)\} \rightsquigarrow D(a) \quad \{D(a)\} \rightsquigarrow A(a) \quad \{A(a)\} \rightsquigarrow D(a)$$
$$\{A(b)\} \rightsquigarrow B(b) \quad \{B(b)\} \rightsquigarrow C(b)$$

Note that $\mathcal{G}$ is indeed included in the intersection of the Pareto-optimal repairs (which is actually equal to $\mathcal{G}$ here).

2.2 Answer Set Programming with Quantifiers

We now recall basic notions about Answer Set Programming (ASP) and ASP with Quantifiers (ASP(Q)).

ASP In ASP, *atoms* take the form $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ where $\mathtt{p}$ is a predicate of arity $n \geq 0$ and each *term* $\mathtt{t}_i$ is either a constant (integer or alphanumeric string starting with lowercase letter) or a variable (alphanumeric string starting with uppercase letter). A *literal* is an atom a or its negation $\mathtt{not}\ a$, where $\mathtt{not}$ represents *negation as failure*. It is *negative* if it is of the form $\mathtt{not}\ a$, *positive* otherwise. An *ASP program*[1] is a finite set of *rules* of the form h :- $l_1, \ldots, l_n$ (with $n \geq 0$), whose *head* h is an atom, and whose *body* $l_1, \ldots, l_n$ is interpreted as the conjunction of the literals $l_1, \ldots, l_n$. Every rule must be *safe*, i.e., each variable appearing in it appears in some positive body literal. A *constraint* is a rule with an empty head (:- $l_1, \ldots, l_n$), and a *fact* is a rule with an empty body (h :-). We also allow *choice rules* of the forms $\{h\}$:- $l_1, \ldots, l_n$ and $1\{h_1; h_2\}1$:- $l_1, \ldots, l_n$: the former is used to choose to either add or omit h, and the latter enforces that precisely one of h_1 and h_2 holds, when the rule body is satisfied. Choice rules are syntactic sugar which do not increase the expressive power but enable compact and intuitive modeling (Calimeri et al. 2020). An ASP expression (atom, rule, program, etc.) is *ground* if it contains no variable.

Given an ASP program P, the *Herbrand Universe* of P is the set $\mathcal{U}_P$ of constants appearing in P and the *Herbrand Base* of P is the set $\mathcal{B}_P$ of ground atoms constructed from predicates of P and constants in $\mathcal{U}_P$. We denote by $\mathit{ground}(P)$ the set of all possible ground rules obtained from rules in P by proper variable substitution with constants in $\mathcal{U}_P$ (cf. (Calimeri et al. 2020)). An *interpretation* is a set of atoms $I \subseteq \mathcal{B}_P$. A positive (resp. negative) ground literal $l = a$ (resp. $l = \mathtt{not}\ a$) is true w.r.t. an interpretation I if $a \in I$ (resp. $a \notin I$), and false otherwise. A conjunction of ground literals is true w.r.t. I if all the literals are true w.r.t. I, and false otherwise. An interpretation I satisfies a ground rule r if the body of r is false or its head is true w.r.t. I. It is a *model* of an ASP program P if it satisfies every $r \in \mathit{ground}(P)$. The *GL-reduct* (Gelfond and Lifschitz 1991) of P w.r.t. I is the program P^I obtained from $\mathit{ground}(P)$ by (i) removing each rule having at least one negative body literal false w.r.t. I; and (ii) removing negative literals from the remaining rules. An *answer set* of P is an interpretation I such that I is a subset-minimal model of P^I. We denote by $AS(P)$ the set of all answer sets of P. A program P is *coherent* if it has some answer set.

[1] We consider core ASP, without disjunction in rule heads.

ASP(Q) ASP with quantifiers (ASP(Q)) (Amendola, Ricca, and Truszczynski 2019) extends ASP by allowing quantification over answer sets of different ASP programs. An *ASP(Q) program* is an expression of the form:

$$\Box_1 P_1 \dots \Box_n P_n : C \tag{1}$$

where C is a stratified[2] ASP program (Ceri, Gottlob, and Tanca 1990) with constraints and for each $i \in \{1, \dots, n\}$, $\Box_i \in \{\exists^{st}, \forall^{st}\}$ is a quantifier and P_i is an ASP program. Given an ASP(Q) program Π of form (1), an ASP program P, and an interpretation I, define the following set of facts and constraints and ASP(Q) program, respectively:

$$\begin{aligned} \mathit{fix}_P(I) =& \{a \text{ :- } \mid a \in \mathcal{B}_P \cap I\} \cup \{ \text{ :- } a \mid a \in \mathcal{B}_P \setminus I\} \\ \Pi_{P,I} =& \Box_1 P_1 \cup \mathit{fix}_P(I) \dots \Box_n P_n : C \end{aligned}$$

The ASP(Q) semantics is defined inductively:

- $\exists^{st} P : C$ is coherent if and only if there exists $M \in AS(P)$ such that $C \cup \mathit{fix}_P(M)$ is coherent;
- $\forall^{st} P : C$ is coherent if and only if for each $M \in AS(P)$, $C \cup \mathit{fix}_P(M)$ is coherent;
- $\exists^{st} P\, \Pi$ is coherent if and only if there exists $M \in AS(P)$ such that $\Pi_{P,M}$ is coherent;
- $\forall^{st} P\, \Pi$ is coherent if and only if for each $M \in AS(P)$, $\Pi_{P,M}$ is coherent.

The *quantified answer sets* of an existential ASP(Q) program $\exists^{st} P\, \Pi$, with Π of form (1), are all $M \in AS(P)$ such that $\Pi_{P,M}$ is coherent.

Example 2. *Let* $\Pi = \exists^{st} P_1 \forall^{st} P_2 : C$ *where:*

$$P_1 = \left\{ \begin{array}{l} \texttt{a :- not b} \\ \texttt{b :- not a} \\ \texttt{c :- not d} \\ \texttt{d :- not c} \end{array} \right\} \quad P_2 = \left\{ \begin{array}{l} \texttt{e :- a, not f} \\ \texttt{f :- a, not e} \\ \texttt{:- e, d} \\ \texttt{:- f, d} \end{array} \right\}$$

$$C = \{ \text{ :- f } \}$$

Let us check whether Π *is coherent, i.e., whether there is an answer set* M_1 *of* P_1 *such that* $\forall^{st} P_2 \cup \mathit{fix}_{P_1}(M_1) : C$ *is coherent. We have* $AS(P_1) = \{\{\texttt{a}, \texttt{c}\}, \{\texttt{a}, \texttt{d}\}, \{\texttt{b}, \texttt{c}\}, \{\texttt{b}, \texttt{d}\}\}$. *Consider first* $M_1 = \{\texttt{a}, \texttt{c}\}$ *and let* $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$. P_2' *has two answer sets:* $M_2' = \{\texttt{a}, \texttt{c}, \texttt{e}\}$ *and* $M_2'' = \{\texttt{a}, \texttt{c}, \texttt{f}\}$, *so for* $\forall^{st} P_2' : C$ *to be coherent, both* $C \cup \mathit{fix}_{P_2'}(M_2')$ *and* $C \cup \mathit{fix}_{P_2'}(M_2'')$ *have to be coherent. However,* $C \cup \mathit{fix}_{P_2'}(M_2'')$ *is incoherent (it contains both* `:- f` *and* `f :-` *). Thus,* $M_1 = \{\texttt{a}, \texttt{c}\}$ *is not a quantified answer set of* Π *and so not a witness for the coherence of* Π. *Consider now* $M_1 = \{\texttt{b}, \texttt{c}\}$. *Then* $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$ *has exactly one answer set,* $M_2 = \{\texttt{b}, \texttt{c}\}$. *In this case* $C \cup \mathit{fix}_{P_2'}(M_2)$ *is coherent as* `f` *is false w.r.t.* M_2. *Thus,* $M_1 = \{\texttt{b}, \texttt{c}\}$ *is a quantified answer set of* Π *and* Π *is coherent.*

Let us now consider Π' *of the form* $\exists^{st} P_1 \forall^{st} P_2 : C'$ *where* $C' = \{$ `:- not fail`$\}$. *Observe that* `fail` *does not appear in any rule head of* P_1, P_2, *and* C'. *Hence,* $C' \cup \mathit{fix}_{P_2 \cup \mathit{fix}_{P_1}(M_1)}(M_2)$ *will be incoherent for every choice of* $M_1 \in AS(P_1)$ *and* $M_2 \in AS(P_2 \cup \mathit{fix}_{P_1}(M_1))$, *since* `fail` *is false w.r.t. any answer set of these programs. Nonetheless, this does not imply that* Π' *is incoherent as well. Indeed,* $M_1 = \{\texttt{a}, \texttt{d}\}$ *is a quantified answer set of* Π' *because* $P_2' = P_2 \cup \mathit{fix}_{P_1}(M_1)$ *is incoherent, so the universal quantification over answer sets of* P_2' *is trivially satisfied.*

[2]Stratified programs respect some conditions that prevent default negation to be involved in recursion.

3 Encoding Semantics in ASP(Q)

In this section, we present our approach to compute the answers that hold under the considered optimal repair-based semantics using ASP and ASP(Q) from their causes, the conflicts and the priority relation. Following Bienvenu et al. (2025), we assume that the input is given by ASP facts on the following predicates: `conf` and `cause` are unary predicates that store identifiers of the KB conflicts $\mathit{Conf}(\mathcal{K})$ and query answer causes $\mathit{Causes}(q(\vec{a}), \mathcal{K})$, respectively, `inConf` and `inCause` are binary predicates such that `inConf(C,A)` (resp. `inCause(C,A)`) means that the fact with identifier `A` belongs to the conflict (resp. cause) with identifier `C`, and `pref` is a binary predicate such that `pref(A,B)` means that $\alpha \succ \beta$, where α and β are the facts with identifiers `A` and `B` respectively.

3.1 Naive Encodings

The logic programs we assemble to build ASP(Q) encodings for semantics based on optimal repairs are given in Table 1.

G-Brave and G-AR Using the programs from Table 1, we build the following ASP(Q) programs for G-brave and G-AR semantics (where $\Pi_1\Pi_2$ stands for $\Pi_1 \cup \Pi_2$):

$$\begin{aligned} \Pi^G_{brave} =& \exists^{st} \Pi_{ReachAll} \Pi_{Rep} \Pi_{SomeCause} \forall^{st} \Pi_{GImp} : C \\ \Pi^G_{AR} =& \exists^{st} \Pi_{ReachAll} \Pi_{Rep} \Pi_{NoCause} \forall^{st} \Pi_{GImp} : C \end{aligned}$$

with $C = \{$ `:- not fail`$\}$.

Let us first explain how Π^G_{brave} works. We want to check whether there exists $\mathcal{R} \in \mathit{GRep}(\mathcal{K}_\succ)$ such that $\mathcal{C} \subseteq \mathcal{R}$ for some $\mathcal{C} \in \mathit{Causes}(q(\vec{a}), \mathcal{K})$. First, applying the existential quantifier over $\Pi_{ReachAll} \cup \Pi_{Rep} \cup \Pi_{SomeCause}$ serves to search for the existence of a repair that contains a cause. Since the facts that do not appear in any conflict belong to every repair, in what follows, we do not distinguish between actual repairs of $\mathcal{K}$ and repairs of $(\mathcal{D}', \mathcal{T})$, where $\mathcal{D}'$ contains all facts that occur in some conflict or some cause (i.e., facts whose identifiers are mentioned in our input ASP programs).

- The rules in $\Pi_{ReachAll}$ compute the set of all such facts, encoded as atoms of the form `reachable(A)`, where `A` is a fact identifier (we will see next how to restrict this set of facts using some notions of reachability, hence the name).
- The rules in Π_{Rep} compute a repair $\mathcal{R}$ of the set of facts whose identifiers appear in the `reachable(A)` atoms. The rules in $\Pi_{SubRep} \subseteq \Pi_{Rep}$ compute a $\mathcal{T}$-consistent set $\mathcal{R}$ of facts as follows. The choice rule in Π_{SubRep} guesses for each relevant fact α with identifier `A` whether $\alpha \in \mathcal{R}$, which is encoded as `inRepair(A)`. Remaining rules in Π_{SubRep} enforce $\mathcal{T}$-consistency: `solved(C)` is derived if `C` is the identifier of a conflict which contains a fact α with identifier `A` such that $\alpha \notin \mathcal{R}$, so the constraint `:- conf(C), not solved(C)` imposes that

$\Pi_{ReachAll}$	`reachable(A) :- inCause(C,A).` `reachable(A) :- inConf(C,A).`
Π_{Attack}	`non_attacking(C,A) :- conf(C), inConf(C,A), inConf(C,B), A != B, pref(A,B).` `attacks(C,A) :- conf(C), inConf(C,A), not non_attacking(C,A).`

Π_{ReachS}	Π_{Attack}
	`reachable(A) :- cause(C), inCause(C,A).`
	`reachable(A) :- conf(C), attacks(C,B), reachable(B), inConf(C,A).`

Π_{ReachW}	`weak_attacks(C,A) :- conf(C), inConf(C,A), inConf(C,B), A != B, not pref(A,B).`
	`reachable(A) :- cause(C), inCause(C,A).`
	`reachable(A) :- conf(C), weak_attacks(C,B), reachable(B), inConf(C,A).`

$\Pi_{ReachBin}$	`reachable(A) :- cause(C), inCause(C,A).` `reachable(A) :- conf(C), inConf(C,B), reachable(B), inConf(C,A), not pref(B,A).`

Π_{SubRep}	`{inRepair(A)} :- reachable(A).`
	`solved(C) :- inConf(C,A), not inRepair(A).`
	`:- conf(C), not solved(C).`

Π_{Rep}	Π_{SubRep}
	`safe(C) :- inConf(C,A), inConf(C,B), not A = B, not inRepair(A), not inRepair(B).`
	`keepOut(A) :- inConf(C,A), not inRepair(A), not safe(C).`
	`:- reachable(A), not inRepair(A), not keepOut(A).`

$\Pi_{SatIfCause}$	`violatedCause(C) :- inCause(C,A), not inRepair(A).` `sat :- cause(C), not violatedCause(C).`
$\Pi_{SomeCause}$	$\Pi_{SatIfCause}$ `:- not sat.`
$\Pi_{NoCause}$	$\Pi_{SatIfCause}$ `:- sat.`

Π_{GImp}	`{global_imp(A)} :- reachable(A).`
	`solved_global(C) :- inConf(C,A), not global_imp(A).`
	`:- conf(C), not solved_global(C).`
	`impMinusRepair(A) :- global_imp(A), not inRepair(A).`
	`repairMinusImp(A) :- inRepair(A), not global_imp(A).`
	`diff :- impMinusRepair(A).`
	`diff :- repairMinusImp(A).`
	`fake_improvement :- not diff.`
	`ok(A) :- repairMinusImp(A), impMinusRepair(B), pref(B,A).`
	`fake_improvement :- repairMinusImp(A), not ok(A).`
	`global_improvement :- not fake_improvement.`
	`:- not global_improvement.`

Π_{POpt}	Π_{Attack}
	`valid(A) :- reachable(A), inRepair(A).`
	`invalid_att(C,A) :- reachable(A), attacks(C,A), inConf(C,B), not inRepair(B), not A = B.`
	`valid(A) :- reachable(A), conf(C), not inRepair(A), attacks(C,A), not invalid_att(C,A).`
	`:- reachable(A), not valid(A).`

$\Pi_{POptBin}$	`valid(A) :- reachable(A), inRepair(A).`
	`valid(A) :- reachable(A), conf(C), not inRepair(A), inConf(C,A), not pref(A,B), inConf(C,B), inRepair(B).`
	`:- reachable(A), not valid(A).`

Π_{Compl}	`pref_comp(A,B) :- reachable(A), reachable(B), pref(A,B).`
	`1{pref_comp(A,B); pref_comp(B,A)}1 :- reachable(A), reachable(B), inConf(C,A), inConf(C,B), not pref(A,B), not pref(B,A), not A = B.`
	`trans_cl_comp(A,B) :- pref_comp(A,B).`
	`trans_cl_comp(A,B) :- trans_cl_comp(A,Y), pref_comp(Y,B).`
	`:- trans_cl_comp(A,A).`

Π_{COpt}	Π_{Compl}
	`valid(A) :- reachable(A), inRepair(A).`
	`invalid_att(C,A) :- reachable(A), not inRepair(A), inConf(C,A), not A = B, inConf(C,B), not inRepair(B).`
	`invalid_att(C,A) :- reachable(A), not inRepair(A), inConf(C,A), inConf(C,B), not A = B, pref_comp(A,B).`
	`valid(A) :- reachable(A), not inRepair(A), inConf(C,A), not invalid_att(C,A).`
	`:- reachable(A), not valid(A).`

Table 1: Logic programs used to built the ASP(Q) programs that filter query answers that hold under X-brave or X-AR semantics from facts on predicates `conf`, `inConf`, `pref`, `cause` and `inCause`. Intuitively, variables `A`, `B` are intended for some fact identifiers, and `C` for some conflict or cause identifier.

at least one fact from each conflict is excluded. The rules in $\Pi_{Rep} \setminus \Pi_{SubRep}$ ensure $\subseteq$-maximality. Specifically, they identify each conflict $\mathcal{C}$ with identifier `C` that contains at least two facts not in $\mathcal{R}$, encoded by `safe(C)`, and for every non-safe conflict $\mathcal{C}$, they mark the only fact $\alpha \in \mathcal{C} \setminus \mathcal{R}$ with identifier `A` using `keepOut(A)`. The last constraint imposes that every $\alpha \notin \mathcal{R}$ is marked by `keepOut`, meaning that there is a conflict $\mathcal{C}$ such that $\alpha \in \mathcal{C}$ and $\mathcal{C} \setminus \{\alpha\} \subseteq \mathcal{R}$, i.e., $\mathcal{R} \cup \{\alpha\}$ is $\mathcal{T}$-inconsistent.

- The rules in $\Pi_{SomeCause}$ ensure $\mathcal{R}$ contains some cause. The rules in $\Pi_{SatIfCause}$ derive `violatedCause(C)` for every cause $\mathcal{C}$ with identifier `C` such that $\mathcal{C} \not\subseteq \mathcal{R}$, and the atom `sat` is derived if there exists some cause $\mathcal{C}$ for which `violatedCause(C)` is not derived (hence $\mathcal{C} \subseteq \mathcal{R}$). The constraint in $\Pi_{SomeCause} \setminus \Pi_{SatIfCause}$ imposes that `sat` is true, so at least one cause must be included.

Hence, an answer set of $\Pi_{ReachAll} \cup \Pi_{Rep} \cup \Pi_{SomeCause}$ corresponds to some $\mathcal{R} \in SRep(\mathcal{K})$ with $(\mathcal{R}, \mathcal{T}) \models q(\vec{a})$. The second program of Π^G_{brave}, Π_{GImp}, is then used to search for a global improvement of $\mathcal{R}$.

- The three first rules guess a $\mathcal{T}$-consistent set of facts $\mathcal{B}$, in the same way as Π_{SubRep} did, using `global_imp` to store the identifiers of the facts in $\mathcal{B}$. The remaining rules verify whether $\mathcal{B}$ is indeed a global improvement of $\mathcal{R}$.
- Two rules compute $\mathcal{B} \setminus \mathcal{R}$ and $\mathcal{R} \setminus \mathcal{B}$, using predicates `impMinusRepair` and `repairMinusImp`, respectively. This is used to (i) check whether $\mathcal{B} \neq \mathcal{R}$ with the two rules that derive `diff` if either $\mathcal{B} \setminus \mathcal{R} \neq \emptyset$ or $\mathcal{R} \setminus \mathcal{B} \neq \emptyset$ and (ii) check whether for every $\alpha \in \mathcal{R} \setminus \mathcal{B}$, there exists $\beta \in \mathcal{B} \setminus \mathcal{R}$ such that $\beta \succ \alpha$ with the rule that derives `ok(A)` if $\alpha \in \mathcal{R} \setminus \mathcal{B}$ with identifier `A` satisfies this condition. The atom `fake_improvement` is derived when (i) or (ii) is not satisfied, i.e., if `diff` is false or if `ok(A)` is false for the identifier `A` of some $\alpha \in \mathcal{R} \setminus \mathcal{B}$. Hence, if `fake_improvement` is false, $\mathcal{B}$ is a global improvement of $\mathcal{R}$, and `global_improvement` is derived.
- The final constraint enforces that `global_improvement` has been derived, so that $\mathcal{B}$ is a global improvement of $\mathcal{R}$.

Finally, C contains only the constraint `:- not fail` and `fail` does not appear in the head of any rule. Hence, as explained in Example 2, if we let $P_1 = \Pi_{ReachAll} \cup \Pi_{Rep} \cup \Pi_{SomeCause}$ and $P_2 = \Pi_{GImp}$, a quantified answer set M of Π^G_{brave} must be an answer set of P_1 such that $P_2 \cup fix_{P_1}(M)$ is incoherent. The next proposition follows.

Proposition 1. *$\mathcal{K}_\succ \models^G_{\text{brave}} q(\vec{a})$ iff Π^G_{brave} is coherent.*

The program for G-AR, Π^G_{AR}, is exactly as Π^G_{brave} except that $\Pi_{SomeCause}$ is replaced by $\Pi_{NoCause}$, which ensures that the repair $\mathcal{R}$ built by $\Pi_{ReachAll} \cup \Pi_{Rep}$ does not contain any cause for $q(\vec{a})$. Indeed, `:- sat` $\in \Pi_{NoCause}$ requires that `sat` is not derived by $\Pi_{SatIfCause}$, so that there is no cause $\mathcal{C}$ such that $\mathcal{C} \subseteq \mathcal{R}$.

Proposition 2. *$\mathcal{K}_\succ \models^G_{\text{AR}} q(\vec{a})$ iff Π^G_{AR} is incoherent.*

We also considered an alternative program of the form $\forall^{st} P \exists^{st} P' : C'$ for G-AR, but as it was less efficient in practice, we present it only in the appendix of the extended version (Bienvenu et al. 2026).

P- and C- Brave and AR For $X \in \{P, C\}$, since the data complexity of query answering under X-brave and X-AR semantics is in the first level of the polynomial hierarchy, we use the following plain ASP programs, which verify whether there exists an optimal repair that either contains some cause or does not contain any cause for the query.

$$\Pi^X_{brave} = \Pi_{ReachAll} \cup \Pi_{SubRep} \cup \Pi_{XOpt} \cup \Pi_{SomeCause}$$
$$\Pi^X_{AR} = \Pi_{ReachAll} \cup \Pi_{SubRep} \cup \Pi_{XOpt} \cup \Pi_{NoCause}$$

As explained in the case $X = G$, $\Pi_{ReachAll} \cup \Pi_{SubRep}$ guesses a $\mathcal{T}$-consistent set $\mathcal{R}$ of facts. The programs Π_{POpt} and Π_{COpt} ensure that $\mathcal{R}$ is a Pareto- or completion-optimal repair, respectively, and were used by Bienvenu et al. (2025).

- The rules in $\Pi_{Attack} \subseteq \Pi_{POpt}$ compute the attack relation $\rightsquigarrow$. They produce `attacks(C, A)` if `C` and `A` are identifiers of a conflict $\mathcal{C}$ and fact α such that $\mathcal{C} \setminus \{\alpha\} \rightsquigarrow \alpha$. The next rules in Π_{POpt} derive `valid(A)` for the identifier `A` of a fact α if (i) $\alpha \in \mathcal{R}$, or (ii) α is attacked by $\mathcal{C} \setminus \{\alpha\}$ for some conflict $\mathcal{C}$ such that $\mathcal{C} \setminus \{\alpha\} \subseteq \mathcal{R}$. The constraint `:- reachable(A), not valid(A)` thus ensures that every $\alpha \notin \mathcal{R}$ is attacked by a set of facts included in $\mathcal{R}$. This means that $\mathcal{R}$ is $\subseteq$-maximal and Pareto-optimal: otherwise, there would be $\alpha \notin \mathcal{R}$ such that $\mathcal{R} \cup \{\alpha\} \setminus \{\beta \mid \alpha \succ \beta\}$ is $\mathcal{T}$-consistent and α would not be attacked by any subset of $\mathcal{R}$.
- The rules in $\Pi_{Compl} \subseteq \Pi_{COpt}$ guess a completion $\succ'$ of $\succ$, using `pref_comp(A, B)` to indicate that $\alpha \succ' \beta$, where α and β have identifiers `A` and `B`. The next rules in Π_{COpt} derive `valid(A)` for the identifier `A` of a fact α if (i) $\alpha \in \mathcal{R}$, or (ii) α is in a conflict $\mathcal{C}$ such that $\mathcal{C} \setminus \{\alpha\} \subseteq \mathcal{R}$ and $\alpha \not\succ' \beta$ for every $\beta \in \mathcal{C} \setminus \{\alpha\}$. The constraint `:- reachable(A), not valid(A)` thus ensures that every $\alpha \notin \mathcal{R}$ is attacked (w.r.t. the attack relation defined w.r.t. $\succ'$) by a set of facts included in $\mathcal{R}$.

Finally, as before, $\Pi_{SomeCause}$ (resp. $\Pi_{NoCause}$) enforces that $\mathcal{R}$ contains a cause (resp. does not contain any cause).

Proposition 3. *For $X \in \{P, C\}$, $\mathcal{K}_\succ \models^X_{\text{brave}} q(\vec{a})$ iff Π^X_{brave} is coherent, and $\mathcal{K}_\succ \models^X_{\text{AR}} q(\vec{a})$ iff Π^X_{AR} is incoherent.*

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 X-IAR.

3.2 Localization

To avoid considering the whole dataset in the encodings, we *localize* them to relevant facts, defined from the query causes using some notions of *reachability* w.r.t. the conflicts and priority relation. We define two such notions of reachability (strong and weak). The first one was already used to localize SAT or ASP encodings by Bienvenu and Bourgaux (2022) and Bienvenu et al. (2025), and the proofs of the theorems below are strongly inspired by the proofs of correctness of the SAT encodings for non-binary conflicts given in the extended version of (Bienvenu and Bourgaux 2022).

Definition 3. *Given a prioritized KB $\mathcal{K}_\succ$ with $\mathcal{K} = (\mathcal{D}, \mathcal{T})$ in $\mathcal{K}_\succ$ and $\mathcal{B} \subseteq \mathcal{D}$, $\mathsf{R_s}(\mathcal{B})$ is the set of facts reachable from*

$\Pi_{\Gamma(\emptyset)}$	`unsafe(A,0) :- attacks(C,A).` `safe(A,0) :- inConf(C,A), not unsafe(A,0).`

$\Pi_{\Gamma incr}$	`safe(A,t) :- safe(A,t − 1).`
	`non_subset(C,A,t) :- conf(C), inConf(C,A), inConf(C,B), A != B, not safe(B,t − 1).`
	`subset(C,A,t) :- conf(C), inConf(C,A), not non_subset(C,A,t).`
	`protected(C,A,t) :- attacks(C,A), inConf(C,B), A != B, attacks(C2,B), subset(C2,B,t).`
	`unsafe(A,t) :- attacks(C,A), not protected(C,A,t).`
	`safe(A,t) :- inConf(C,A), not unsafe(A,t).`
	`continue(t) :- safe(A,t), not safe(A,t − 1).`

Table 2: Logic programs used to compute the grounded repair from facts on predicates `conf`, `inConf`, and `attacks` (computed by Π_{Attack}).

$\mathcal{B}$ in the directed hypergraph[3] *whose edges are* $\{(\alpha, \mathcal{E}) \mid \mathcal{E} \rightsquigarrow \alpha\} = \{(\alpha, \mathcal{E}) \mid \mathcal{E} \cup \{\alpha\} \in Conf(\mathcal{K}), \forall \beta \in \mathcal{E}, \alpha \not\succ \beta\}$ *and* $\mathsf{R_w}(\mathcal{B})$ *is the set of facts reachable from $\mathcal{B}$ in the directed hypergraph whose edges are* $\{(\alpha, \mathcal{E}) \mid \mathcal{E} \cup \{\alpha\} \in Conf(\mathcal{K}), \exists \beta \in \mathcal{E}, \alpha \not\succ \beta\}$.

Note that $\mathsf{R_s}(\mathcal{B}) \subseteq \mathsf{R_w}(\mathcal{B})$ and that the these two sets coincide when the conflicts are *binary*: in this case, reachability is done in the oriented graph whose edges are $\{(\alpha, \beta) \mid \{\alpha, \beta\} \in Conf(\mathcal{K}), \alpha \not\succ \beta\}$.

Theorem 3. *Let* $X \in \{P, G, C\}$, $\mathcal{B} \subseteq \mathcal{D}$, $\mathcal{B}_r = \mathsf{R_w}(\mathcal{B})$, $\mathcal{K}^{\mathcal{B}_r} = (\mathcal{B}_r, \mathcal{T})$ *and* $\succ^{\mathcal{B}_r}$ *be the restriction of* $\succ$ *to* $\mathcal{B}^r$.

- *If* $\mathcal{R} \in XRep(\mathcal{K}_\succ)$, *then* $\mathcal{R} \cap \mathcal{B}_r \in XRep(\mathcal{K}^{\mathcal{B}_r}_{\succ^{\mathcal{B}_r}})$.
- *If* $\mathcal{R} \in XRep(\mathcal{K}^{\mathcal{B}_r}_{\succ^{\mathcal{B}_r}})$, *then there exists* $\mathcal{R}' \in XRep(\mathcal{K}_\succ)$ *such that* $\mathcal{R} = \mathcal{R}' \cap \mathcal{B}_r$.

Theorem 4. *If* $\mathcal{B}_r = \mathsf{R_w}(\mathcal{B})$ *is replaced by* $\mathcal{B}_r = \mathsf{R_s}(\mathcal{B})$, *Theorem 3 still holds for* $X \in \{P, C\}$.

We can thus replace $\Pi_{ReachAll}$ by Π_{ReachW} or Π_{ReachS} (if $X \in \{P, C\}$) in the naive encodings, in order to restrict the set of facts from $\mathcal{D}$ considered (whose identifiers are stored in `reachable`). Indeed, these programs compute $\mathsf{R_s}(\mathcal{B})$ and $\mathsf{R_w}(\mathcal{B})$ for $\mathcal{B} = \bigcup_{\mathcal{C} \in Causes(q(\vec{a}), \mathcal{K})} \mathcal{C}$, respectively. We leave open whether Theorem 4 holds for $X = G$.

3.3 Simplified Encoding for Binary Conflicts

When the conflicts are of size exactly 2 (we assume that any self-inconsistent fact has been removed from the dataset), we can simplify the encodings as follows.

Reachable Facts First, as explained earlier, the set of relevant (`reachable`) facts is easier to compute with binary conflicts: we replace Π_{ReachW} or Π_{ReachS} by $\Pi_{ReachBin}$.

Repairs In the encodings for semantics based on globally-optimal repairs, which build a full, $\subseteq$-maximal, repair, we modify Π_{Rep} by replacing the two rules of Π_{Rep} that compute the facts that are necessary to keep out of the repair by the simpler rules (with one negation instead of four):

```
dangerous(C) :- conf(C), inConf(C,A), inRepair(A).
keepOut(A) :- dangerous(C), inConf(C,A),
              not inRepair(A).
```

[3] A node γ is reachable from a set of nodes $\mathcal{B}$ in a directed hypergraph if $\gamma \in \mathcal{B}$ or there exists a node β reachable from $\mathcal{B}$ and some edge $(\beta, \mathcal{E})$ such that $\gamma \in \mathcal{E}$.

Pareto-Optimality We use again the fact that when the conflicts are binary the attack relation is straightforward ($\{\beta\} \rightsquigarrow \alpha$ iff $\{\alpha, \beta\} \in Conf(\mathcal{K})$ and $\alpha \not\succ \beta$), and that including a single conflicting fact is sufficient to exclude a fact, to replace Π_{POpt} by $\Pi_{POptBin}$.

3.4 Approximations

Another way to answer queries under optimal repair-based semantics more efficiently is to use *approximations* to compute subsets or supersets of the answers and thus avoid relying on more demanding programs for some answers.

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 2.1) and the *trivially P-IAR answers* (Bienvenu, Bourgaux, and Goasdoué 2014; Bienvenu and Bourgaux 2022), which have some cause such that none of its fact is attacked (w.r.t. $\rightsquigarrow$). Note that the trivially P-IAR answers are those that hold w.r.t. $\Gamma(\emptyset)$. Table 2 shows the ASP programs used to compute $\Gamma(\emptyset)$ then incrementally compute the grounded repair. Using incremental solving mirrors the fixpoint definition of grounded semantics and allows for a very direct encoding. Once we have computed one of these two sets, we use it to filter the answers that have some cause included in it.

P-AR and P-Brave We also investigate the use of semantics whose data complexity is in the first level of the polynomial hierarchy, e.g., based on Pareto-optimal repairs, to obtain lower or upper bounds on semantics based on globally-optimal repairs: $\mathcal{K}_\succ \models^P_{\text{AR}} q(\vec{a})$ implies $\mathcal{K}_\succ \models^G_{\text{AR}} q(\vec{a})$ (hence also $\mathcal{K}_\succ \models^G_{\text{brave}} q(\vec{a})$) and $\mathcal{K}_\succ \not\models^P_{\text{brave}} q(\vec{a})$ implies $\mathcal{K}_\succ \not\models^G_{\text{brave}} q(\vec{a})$ (hence also $\mathcal{K}_\succ \not\models^G_{\text{AR}} q(\vec{a})$).

4 Experiments

Our experimental evaluation aims at (i) evaluating the impact of adopting globally-optimal repairs, (ii) assessing the grounded semantics, and (iii) comparing the different approaches of the same problem in terms of runtime. In more detail, we consider the following questions:

- What proportion of the intersection of optimal repairs is given by the grounded repair? And by the trivially P-IAR facts (i.e., $\Gamma(\emptyset)$)? What is the overhead in term of runtime to compute the grounded repair instead of $\Gamma(\emptyset)$?

		$\cap_S$	$\Gamma^1 \setminus \cap_S$	$\Gamma^2 \setminus \Gamma^1$	$\Gamma^3 \setminus \Gamma^2$	$\Gamma^4 \setminus \Gamma^3$
u1c1	$\succ^{ns}$	73351	921	1365	0	0
u1c50	$\succ^{ns}$	43779	7289	21319	1329	8
u1c1	$\succ^{ss}$	73351	1225	881	0	0
u1c50	$\succ^{ss}$	43779	8493	9368	11	0
u1c1	$\succ^{nb}$	73131	989	1406	4	0
u1c50	$\succ^{nb}$	43612	10777	18629	52	0

Table 3: Number of facts in the different parts of the grounded repair: $\cap_S = \bigcap_{\mathcal{R} \in SRep(\mathcal{K})} \mathcal{R}$, $\Gamma^i = \Gamma^i(\emptyset)$.

	$\succ^{ss}$		$\succ^{ns}$	
	$\mathcal{G} \setminus \cap_S$	$\Gamma^1 \setminus \cap_S$	$\mathcal{G} \setminus \cap_S$	$\Gamma^1 \setminus \cap_S$
u1c1	1.58	0.16	2.10	0.16
u1c20	14.92	1.26	21.22	1.39
u1c50	61.80	3.53	126.39	3.80
u5c1	7.85	0.68	10.70	0.68
u5c20	129.31	7.12	240.34	7.11
u5c50	304.11	14.55	oom	14.69
u20c1	23.75	2.41	47.05	2.39
u20c20	oom	36.07	oom	33.13

Table 4: Time (in seconds) to compute the facts that belong to the grounded repair $\mathcal{G}$ or to $\Gamma^1 = \Gamma(\emptyset)$ (among those that belong to some conflict) from the precomputed attack relation.

- Given a semantics, what is the impact of localization or binary conflicts-specific encodings?
- What is the impact of using globally-optimal repairs instead of Pareto- or completion-optimal ones, both in terms of the answers obtained and runtime overhead?

4.1 Experimental Setting

Following Bienvenu et al. (2025), we translated into ASP programs a subset of the ORBITS benchmark (Bienvenu and Bourgaux 2022) which provides several conflict sets, priority relations, and potential answers associated with their causes built from the CQAPri benchmark (Bourgaux 2016), a synthetic benchmark adapted from LUBM$^{\exists}_{20}$ (Lutz et al. 2013) to evaluate inconsistency-tolerant query answering over DL-Lite KBs. To experiment also with non-binary conflicts, Bienvenu et al. (2025) added a denial constraint that yields conflicts of size 10 and built some priority relations for this case using preference rules.

Datasets The datasets of the CQAPri benchmark are named uXcY, with X and Y related to the size and the proportion of facts involved in some conflicts respectively, and are such that uXcY $\subseteq$ uXcY$'$ for Y $\leq$ Y$'$ and uXcY $\subseteq$ uX$'$cY for X $\leq$ X$'$. Since our focus is on the use of the more demanding globally-optimal repairs, we mostly use the smallest datasets u1cY with Y $\in \{1, 5, 10, 20, 30, 50\}$, which contain from 75K to 78K facts. The proportion of facts involved in some (binary) conflict in these datasets ranges from 3% to 44% and the corresponding conflict sets contain from 2K to 81K conflicts, involving 2K to 34K facts.

		$\cap_S$	$\mathcal{G} \setminus \cap_S$	$\cap_P \setminus \mathcal{G}$	$\cap_G \setminus \cap_P$	$\cap_C \setminus \cap_G$
u1c1		73351	2286	0	0	0
u1c10	$\succ^{ns}$	65310	10134	3	22	1
u1c20		56646	18507	82	46	7

Table 5: Size of grounded repair $\mathcal{G}$ and optimal repair intersections: $\cap_X = \bigcap_{\mathcal{R} \in XRep(\mathcal{K}_\succ)} \mathcal{R}$ for $X \in \{S, P, G, C\}$.

Forty non-binary conflicts are generated (in all datasets) when the additional denial constraint is considered. We also run a few experiments with some larger datasets u5cY with Y $\in \{1, 5, 10, 20\}$, with 12K to 231K conflicts involving 12K to 137K facts.

Priority Relations In the binary conflicts case, we use the two priority relations of the ORBITS benchmark: $\succ^{ss}$ is built from priority levels, hence is such that globally-, Pareto- and completion-optimal repairs coincide, while $\succ^{ns}$ is not score-structured and allows us to distinguish between the three kinds of repairs. Moreover, $\succ^{ss}$ assigns a priority between the two facts of about 80% of the conflicts and this proportion is about 60%-65% for $\succ^{ns}$. For the non-binary conflicts case, we use a priority relation $\succ^{nb}$ resulting from some preference rules given by Bienvenu et al. (2025) which assigns a priority between two facts that belong to some conflict in about 90% of the cases.

Queries We use the 8 queries Bienvenu et al. (2025) selected from the CQAPri benchmark for having a lower number of potential answers. The total number of potential answers over all queries ranges from 1,525 (on u1c1) to 8,206 (on u5c20).

Setup All experiments were executed on a machine equipped with an Intel(R) Xeon(R) CPU E7-8880 v4 @ 2.20GHz, running Debian GNU/Linux 12, with memory and CPU (i.e., user+system) limited to 8GB and 600s. Time and memory usage have been measured with pyrunlim[4]. As ASP(Q) solver we used the CASPER system (Andrea, Mazzotta, and Ricca 2026)[5], and as ASP solver we used CLINGO (Gebser et al. 2017).

Preprocessing Reported times exclude the computation of the attack relation using Π_{Attack}, as it can be considered a query-independent preprocessing task. Computing the attack relation took at most 5 seconds for the u1cY datasets (small datasets), about 30 seconds for u5c20 (largest dataset we used for the query answering task) and up to about one hour for u20c50 (dataset with 2M facts, 46% of facts involved in some conflict, and 3M conflicts, which we considered only for the task of computing the grounded repair).

4.2 Experimental Results

In what follows, we summarize our main observations.

[4] pyrunlim is available at https://github.com/alviano/python.git

[5] Other ASP(Q) systems are available, such as QASP (Amendola et al. 2022) and PYQASP (Faber, Mazzotta, and Ricca 2023), but CASPER performed better in our preliminary evaluation.

		Triv	GR\Triv	Pot\GR	P-AR\GR		G-AR\GR		C-AR\GR		P-brave\GR		G-brave\GR		C-brave\GR	
u1c1		1465	59	1	0	(0)	0	(0)	0	(0)	0	(0)	0	(0)	0	(0)
u1c20		937	584	35	1	(0)	0	(10)	1	(0)	17	(0)	16	(6)	17	(0)
u1c50	$\succ^{ns}$	625	875	141	26	(0)	9	(82)	20	(56)	79	(0)	28	(81)	38	(40)
u5c1		7637	316	10	0	(0)	0	(0)	0	(0)	2	(0)	2	(0)	2	(0)
u5c20		4947	3035	224	37	(0)	6	(96)	32	(32)	91	(0)	51	(151)	83	(18)
u1c1		1447	76	2	0	(0)	0	(0)	0	(0)	0	(0)	0	(0)	0	(0)
u1c20	$\succ^{nb}$	940	571	45	0	(0)	0	(2)	0	(0)	1	(0)	0	(2)	1	(0)
u1c50		680	823	138	0	(0)	0	(43)	0	(0)	20	(0)	13	(37)	20	(0)

Table 6: Number of answers found trivially P-IAR (Triv), grounded (GR) but not trivially P-IAR, potential answers (Pot) not grounded, and X-AR or X-brave but not grounded. The number of potential answers for which we ran out of time (600s) is given in parenthesis.

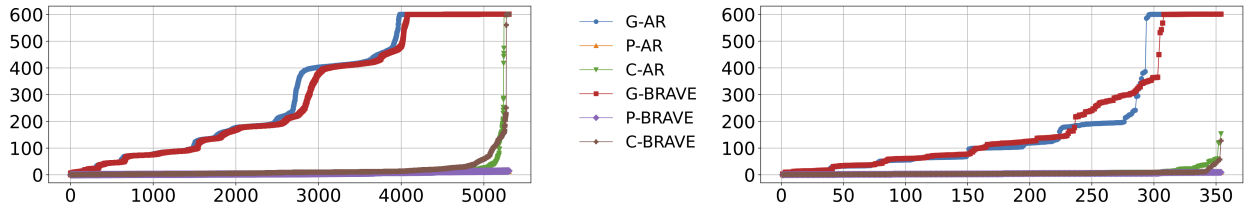

Figure 1: Time (in seconds, y-axis) to decide for each $q(\vec{a})$, dataset uXcY, and priority relation, whether $q(\vec{a})$ holds under X-AR/X-brave, with instances sorted by increasing solving time along the x-axis (i.e., a point (i, j) indicates that i instances could be solved within j seconds). (left) binary conflicts case (u1cY and u5cY, two priority relations) (right) non-binary conflicts case (u1cY, one priority relation)

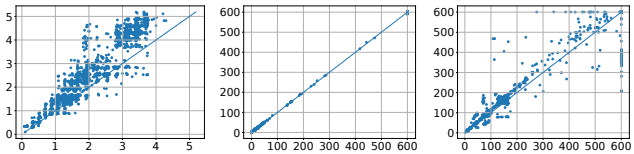

Figure 2: Time for the generic encoding (y-axis) vs time for the encoding specific for binary conflicts (x-axis), both with localization, to decide for each $q(\vec{a})$, dataset uXcY, and priority relation, if $q(\vec{a})$ holds under X-AR. (left) X=P (middle) X=C (right) X=G

Tractable Approximations Table 3 shows the number of plain (S-)IAR facts (not involved in any conflict, $\cap_S$), other trivially P-IAR facts ($\Gamma(\emptyset)$), and facts added at each step of the incremental computation of the grounded repair, for some example datasets and priority relations. Over all the cases we consider, the proportion of facts that belong to the grounded repair but which are not trivially P-IAR (i.e., are in $\mathcal{G} \setminus \Gamma(\emptyset)$) varies from 1%-2% on the datasets with a low proportion of facts in conflicts (uXc1) to 15%-31% (depending on the priority relation) on the dataset with the highest proportion of facts in conflicts u1c50. Table 4 gives examples of times needed to compute the grounded repair and the trivially P-IAR facts. Over all the cases we consider, computing $\mathcal{G}$ instead of $\Gamma(\emptyset)$ from the attack relation multiplies the runtime by up to 34 but it remains below 600s. However, we encounter memory problems for larger datasets (note that for this task we also used datasets u20cY of 2M facts). Table 5 shows the size of the grounded repair and how many facts are added by taking the intersections of the optimal repairs. Given the cost of checking whether each non-grounded fact holds under X-(I)AR, these numbers suggest the interest of using grounded rather than X-IAR for $X \in \{P, G, C\}$. This is also the reason that we did not conduct further experimental evaluations of the X-IAR semantics.

Impact of Localization Unsurprisingly, localization is crucial. For example, on u1c10 with $\succ^{nb}$, we manage to decide for 99% of the pairs of a potential answer (which does not hold under grounded) and a semantics (X-AR or X-brave with $X \in \{P, G, C\}$) whether the answer holds under the semantics, while they all yield a time-out without localization. We adopt localization (using weak reachability for globally-optimal repairs and strong one for Pareto- and completion-optimal repairs) in the rest of the experiments.

Impact of Specific Encoding for Binary Conflicts Figure 2 compares for each potential answer (which does not hold under grounded), dataset with binary conflicts, and priority relation, the time needed to decide if it holds under X-AR semantics with and without using the simplified encodings proposed in the case of binary conflicts. We can see that this simplification is indeed generally helpful when X=P, and mostly neutral when X=C. When X=G, we observe a more diverse outcome: even if the binary conflicts-specific encoding improves the performance in a majority of cases, there are a significant number of cases where it is the opposite. Moreover, the difference between the two encodings is more important (in particular, the two encodings do not lead to time-out on the same instances).

Semantics Comparison in Terms of Runtimes Figure 1 shows the times needed to decide whether a potential answer which does not hold under grounded semantics holds under a given semantics for X-AR and X-brave with $X \in \{P, G, C\}$, over all the cases we consider, using localization and the specific encoding for binary conflicts when possible. Recall that in a cactus plot instances are sorted by solving

time, so a point (i, j) in the plot indicates that for the considered semantics, there were i instances which could be (individually) solved within j seconds. These results confirm that using globally-optimal repairs is significantly more difficult in practice than using Pareto- or even completion-optimal repairs, and the difference becomes more pronounced as the proportion of facts involved in some conflicts increases.

Semantics Comparison in Terms of Answers Table 6 shows the number of answers we obtain under the different semantics and the number of potential (non grounded) answers for which we did not manage to determine whether they hold under the considered semantics in our 600s time limit. A first observation is that the grounded semantics seems to be a very good approximation of the X-AR semantics ($X \in \{P, G, C\}$). This is key to achieving feasibility as it allows us to reduce the number of potential answers for which we need to call the procedures for the P/C-AR/brave semantics (cf. the column Pot\GR which gives the number of potential answers for which we actually need to use the encodings for X-AR or X-brave semantics). For X-brave answers however, we do observe cases where a large share of the answers cannot be obtained without using the designated encodings (in particular in the $\succ^{ss}$ case, which is omitted from the table as in this case the three kinds of optimal repair coincide). Another useful information is that the three kinds of repairs seem to often yield the same answers (recall that answers that hold under P-AR semantics are included in those that hold under G-AR semantics, which are included in those that hold under C-AR semantics, and the inclusions for the X-brave semantics are the other way around).

5 Conclusion

In this paper, we have presented the first implementation of globally-optimal repair-based semantics, using ASP(Q), and of the grounded semantics, making it possible for the first time to compare these semantics with the previously implemented semantics based upon Pareto- and completion-optimal repairs. While we are able to compute query answers for some instances with globally-optimal repairs, the runtimes are significantly longer, with more timeouts, than for the Pareto- and completion-optimal repairs, in line with the worst-case complexity. 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 semantics and P-brave semantics (which are the fastest optimal repair semantics) and only call the G-AR and G-brave procedures if the status remains unresolved.

Our evaluation also highlighted the surprising effectiveness of the grounded semantics as an approximation of the optimal repair-based semantics, an insight which moreover can be applied also to non-ASP-based implementations. Importantly, the grounded repair can be computed as a preprocessing task, thus making it possible to employ a principled inconsistency-tolerant semantics with little overhead compared to standard query answering. This suggests the interest of developing highly optimized methods for computing and updating the grounded repair for very large datasets.

Acknowledgements

This work was supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014); by the Italian Ministry of Industrial Development (MISE) under project EI-TWIN n. F/310168/05/X56 CUP B29J24000680005; and by the Italian Ministry of Research (MUR) under project PNRR FAIR - Spoke 9 - WP 9.1 CUP H23C22000860006.

AI Declaration

The authors have not employed any Generative AI tools.

References

Amendola, G.; Cuteri, B.; Ricca, F.; and Truszczynski, M. 2022. Solving problems in the polynomial hierarchy with ASP(Q). In *Proceedings of LPNMR*.

Amendola, G.; Ricca, F.; and Truszczynski, M. 2019. Beyond NP: quantifying over answer sets. *Theory Pract. Log. Program.* 19(5-6):705–721.

Andrea, C.; Mazzotta, G.; and Ricca, F. 2026. 2-ASP(Q) solving based on CEGAR. In *Proceedings of AAAI*.

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of PODS*.

Bertossi, L. E. 2019. Database repairs and consistent query answering: Origins and further developments. In *Proceedings of PODS*.

Bienvenu, M., and Bourgaux, C. 2020. Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In *Proceedings of KR*.

Bienvenu, M., and Bourgaux, C. 2022. Querying inconsistent prioritized data with ORBITS: algorithms, implementation, and experiments. In *Proceedings of KR*.

Bienvenu, M., and Bourgaux, C. 2023. Inconsistency handling in prioritized databases with universal constraints: Complexity analysis and links with active integrity constraints. In *Proceedings of KR*.

Bienvenu, M., and Rosati, R. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proceedings of IJCAI*.

Bienvenu, M.; Bourgaux, C.; Inoue, K.; and Jean, R. 2025. A rule-based approach to specifying preferences over conflicting facts and querying inconsistent knowledge bases. In *Proceedings of KR*.

Bienvenu, M.; Bourgaux, C.; Jean, R.; and Mazzotta, G. 2026. Using ASP(Q) to handle inconsistent prioritized data. Extended version with appendix available at: https://arxiv.org/abs/2604.21603.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2014. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proceedings of AAAI*.

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

Bourgaux, C. 2016. *Inconsistency Handling in Ontology-Mediated Query Answering. (Gestion des incohérences pour l'accès aux données en présence d'ontologies)*. Ph.D. Dissertation, University of Paris-Saclay, France.

Bourgaux, C. 2025. Inconsistency-tolerant semantics based on (preferred) repairs. In *Proceedings of RW*.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Maratea, M.; Ricca, F.; and Schaub, T. 2020. ASP-Core-2 input language format. *Theory Pract. Log. Program.* 20(2):294–309.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning (JAR)* 39(3):385–429.

Ceri, S.; Gottlob, G.; and Tanca, L. 1990. *Logic Programming and Databases*. Surveys in computer science.

Dixit, A. A., and Kolaitis, P. G. 2019. A SAT-based system for consistent query answering. In *Proceedings of SAT*.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.* 15(3-4):289–323.

Eiter, T.; Fink, M.; Greco, G.; and Lembo, D. 2008. Repair localization for query answering from inconsistent databases. *ACM Trans. Database Syst.* 33(2):10:1–10:51.

Faber, W.; Mazzotta, G.; and Ricca, F. 2023. An efficient solver for ASP(Q). *Theory Pract. Log. Program.* 23(4):948–964.

Faber, W.; Morak, M.; and Chrpa, L. 2022. Determining action reversibility in STRIPS using answer set programming with quantifiers. In *Proceedings of PADL*.

Faber, W. 2024. Solving argumentation problems using answer set programming with quantifiers: Preliminary report. In *Proceedings of ICLP-WS*.

Fandinno, J.; Laferrière, F.; Romero, J.; Schaub, T.; and Son, T. C. 2021. Planning with incomplete information in quantified answer set programming. *Theory Pract. Log. Program.* 21(5):663–679.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2017. Multi-shot ASP solving with clingo. *CoRR* abs/1705.09811.

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

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Gener. Comput.* 9(3/4):365–386.

Greco, G.; Greco, S.; and Zumpano, E. 2003. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.* 15(6):1389–1408.

Kimelfeld, B.; Livshits, E.; and Peterfreund, L. 2017. Detecting ambiguity in prioritized database repairing. In *Proceedings of ICDT*.

Kimelfeld, B.; Livshits, E.; and Peterfreund, L. 2020. Counting and enumerating preferred database repairs. *Theor. Comput. Sci.* 837:115–157.

Kolaitis, P. G.; Pema, E.; and Tan, W. 2013. Efficient querying of inconsistent databases with binary integer programming. *Proc. VLDB Endow.* 6(6):397–408.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant semantics for description logics. In *Proceedings of RR*.

Lutz, C.; Seylan, I.; Toman, D.; and Wolter, F. 2013. The combined approach to OBDA: Taming role hierarchies using filters. In *Proceedings of ISWC*.

Manna, M.; Ricca, F.; and Terracina, G. 2013. Consistent query answering via ASP from different perspectives: Theory and practice. *Theory Pract. Log. Program.* 13(2):227–252.

Staworko, S.; Chomicki, J.; and Marcinkowski, J. 2012. Prioritized repairing and consistent query answering in relational databases. *Annals of Mathematics and Artificial Intelligence (AMAI)* 64(2-3):209–246.