Data Repair of Inconsistent DL-Programs*

Thomas Eiter and **Michael Fink** and **Daria Stepanova**
Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{dasha,eiter,fink}@kr.tuwien.ac.at

Abstract

Nonmonotonic Description Logic (DL) programs support rule-based reasoning on top of Description Logic ontologies, using a well-defined query interface. However, the interaction of the rules and the ontology may cause inconsistency such that no answer set (i.e. model) exists. We thus consider repairing DL-programs, i.e., changing formulas to obtain consistency. Viewing the data part of the ontology as the source of inconsistency, we define program repairs and repair answer sets based on changes to it. We analyze the complexity of the notion, and we extend an algorithm for evaluating DL-programs to compute repair answer sets, under optional selection of preferred repairs. The extension involves a generalized ontology repair problem, in which the entailment and non-entailment of sets of queries with updates to the ontology must be achieved. While this is intractable in general, we identify for the Description Logic $DL\text{-}Lite_{\mathcal{A}}$ some tractable classes of preferred repairs that are useful in practice.

1 Introduction

Nonmonotonic Description Logic (DL-)programs [Eiter *et al.*, 2008b] are a prominent approach to combine Description Logic knowledge bases (alias, ontologies) with nonmonotonic logic programming. Different from other approaches, see [Motik and Rosati, 2010] and references therein, they offer a loose coupling between the rules and the ontology through special *DL-atoms* which serve as query interfaces to the ontology. The possibility to add information from the rules part prior to query evaluation allows for adaptive combinations and to solve advanced reasoning problems on top of ontologies. Noticeably, the abstract design of DL-programs has been fruitfully generalized to model view-based access of external information sources from rules beyond ontologies.

Example 1. Consider the DL-program Π in Figure 1, which captures information about children of a primary school and their parents in simplistic form. It consists of an ontology $\mathcal{O}$

Figure 1: DL-program Π over a family ontology

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ \mathit{Child} \sqsubseteq \exists.\mathit{hasParent} & (4)\ \mathit{Male}(\mathit{pat}) \\ (2)\ \mathit{Adopted} \sqsubseteq \mathit{Child} & (5)\ \mathit{Male}(\mathit{john}) \\ (3)\ \mathit{Female} \sqsubseteq \neg \mathit{Male} & (6)\ \mathit{hasParent}(\mathit{john}, \mathit{pat}) \end{array} \right\}$$

$$P = \left\{ \begin{array}{l} (7)\ \mathit{ischildof}(\mathit{john}, \mathit{alex}); \quad (8)\ \mathit{boy}(\mathit{john}); \\ (9)\ \mathit{hasfather}(X, Y) \leftarrow \mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](Y) \\ \qquad\qquad \mathrm{DL}[; \mathit{hasParent}](X, Y); \\ (10)\ \bot \leftarrow \mathit{not}\ \mathrm{DL}[; \mathit{Adopted}](X), Y_1 \neq Y_2, \\ \qquad \mathit{hasfather}(X, Y_1), \mathit{ischildof}(X, Y_2), \\ \qquad \mathit{not}\ \mathrm{DL}[\mathit{Child} \uplus \mathit{boy}; \neg \mathit{Male}](Y_2); \\ (11)\ \mathit{contact}(X, Y) \leftarrow \mathrm{DL}[; \mathit{hasParent}](X, Y), \\ \qquad\qquad \mathit{not}\ \mathit{omit}(X, Y); \\ (12)\ \mathit{omit}(X, Y) \leftarrow \mathrm{DL}[; \mathit{Adopted}](X), Z \neq Y, \\ \qquad\qquad \mathit{hasfather}(X, Y), \mathit{contact}(X, Z) \end{array} \right\}$$

which contains a taxonomy $\mathcal{T}$ of concepts (i.e., classes) in (1)-(3) and factual data (i.e., assertions) $\mathcal{A}$ about some individuals in (4)-(6). The rules P contain some further facts (7), (8) and proper rules: (9) determines fathers from the ontology, upon feeding information to it; (10) checks, informally, against them for local parent information ($\mathit{ischildof}$) the constraint that a child has for sure at most one father, unless it is adopted; finally (11)-(12) single out contact persons for children, which by default are the parents; for adopted children, fathers from the ontology are omitted if some other contact exists.

However, the information flow between the rules and the ontology can have unforseen effects and cause inconsistency such that no answer set, (i.e., model), of a DL-program exists. In the above example, this happens as john, who is not provably adopted, has pat as father by the ontology, and by the local information possibly also alex.

An inconsistent program yields no information and is unusable. To deal with this, [Pührer *et al.*, 2010; Fink, 2012] effected inconsistency tolerance by suppressing or weaking information leading to inconsistency in model building. However, the problem to *repair* the program, i.e., change formulas in it to obtain consistency (which is a natural and ubiquitous approach in data and knowledge representation), has to the best of our knowledge not been considered.

In this paper, we consider this issue, which due the interaction between the rules and the ontology is nontrivial and at least as challenging as for these parts. Ontology repair has

*This research has been supported by the Austrian Science Fund (FWF) project P20840, P20841, P24090, and by the Vienna Science and Technology Fund (WWTF) project ICT08-020.

been studied in many works, e.g., in [Lembo *et al.*, 2010; Bienvenu, 2012] for consistent query answering; repairing nonmonotonic logic programs instead is less developed (cf. [Sakama and Inoue, 2003]). In the light of this and as the rules are on top of the ontology (such that their plausibility can be separately assessed), we take the view that the latter, and here in particular its data part might not be fully correct.

In the above example, suitable changes of the ontology facts make the DL-program consistent. For example, deleting $hasParent(john, pat)$ from $\mathcal{A}$ leads to the answer set $I_1 = \{ischildof(john, alex),\ boy(john)\}$, while the addition of $Adopted(john)$ leads to $I_2 = \{ischildof(john, alex),\ boy(john),\ hasfather(john, pat), contact(john, pat)\}$; yet other possibilities exist. However, not all repairs might be acceptable; the question is how to find suitable repairs which, as an additional constraint, should not increase the complexity of DL-programs.

We tackle this challenge with the following contributions.

(1) We formalize repairing DL-programs and introduce the notions of repair and repair answer set (Section 3). They are based on changes of the assertions in the ontology that enable answer sets. As it turns out, repair answer sets do not have higher complexity than ordinary answer sets (more precisely, weak and FLP answer sets) if queries in DL-atoms are evaluable in polynomial time; to ensure this, we concentrate on the Description Logic $DL\text{-}Lite_{\mathcal{A}}$ [Calvanese *et al.*, 2007]. We model repair preference by functions σ that select preferred ones from a set of candidates. As preference is a known source of complexity, we focus on selections σ with a benign independence property (which some practicable σ enjoy).

(2) We show how an algorithm that is used for evaluating DL-programs can be gracefully extended to compute repairs resp. repair answer sets, with an optional selection σ featuring independence (Section 4). The extension involves a *generalized ontology repair problem* (ORP), which arises from a candidate answer set and the DL-atoms of the program. It consists of two sets D_1,D_2 of entailment and non-entailment queries to the ontology, with possible temporary assertions, and asks for an ABox satisfying these sets. Importantly, under independence the σ-selected ABoxes also yield, modulo a conditional check on the rules part, the σ-selected repairs of the program.

(3) Furthermore, we analyze the complexity of the ORP problem (Section 5). Unsurprisingly, it is intractable (NP-complete) for $DL\text{-}Lite_{\mathcal{A}}$ in general, but we show that NP-hardness holds also in plain ontology settings, due to the temporary assertions. However, we also identify several tractable cases of σ-selections that are useful in practice.

While we focus here on $DL\text{-}Lite_{\mathcal{A}}$, our approach and basic results can be extended to DL-programs with ontologies in other Description Logics as well.

2 Preliminaries

Informally, a DL-program consists of a description logic knowledge base (or ontology) $\mathcal{O}$ and a set P of logic programming rules that may query $\mathcal{O}$ via special atoms in the rule bodies; the queries are evaluated subject to temporal updates of $\mathcal{O}$ with assertions given by predicate values in P.

2.1 Description Logic Knowledge Bases

We consider Description Logic (DL) knowledge bases (KBs) over a signature $\Sigma_o = \langle \mathbf{I}, \mathbf{C}, \mathbf{R} \rangle$ with a set $\mathbf{I}$ of individuals (constants), a set $\mathbf{C}$ of concept names $\mathbf{C}$ (unary predicates), and a set $\mathbf{R}$ of role names $\mathbf{R}$ (binary predicates) as usual.

A *DL knowledge base* (or *ontology*) is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ of a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$, which are finite sets of formulas capturing taxonomic resp. factual knowledge, whose form depends on the underlying DL.

In $DL\text{-}Lite_{\mathcal{A}}$, concepts C, denoting sets of objects, and roles R, denoting binary relations between objects, are formed according to the following syntax, where $A \in \mathbf{C}$ is an atomic concept and $P \in \mathbf{R}$ an atomic role:

$$C \to A \mid \exists R \qquad R \to P | P^{-}.$$

$DL\text{-}Lite_{\mathcal{A}}$ TBox axioms are then of the form:

$$C_1 \sqsubseteq C_2, \qquad C_1 \sqsubseteq \neg C_2,$$
$$R_1 \sqsubseteq R_2, \qquad R_1 \sqsubseteq \neg R_2, \qquad (func\ R).$$

Axioms in the first column are *positive inclusions* (among concepts and roles, respectively), and those in the second column *disjointness axioms*; $(func\ R)$ is a *functionality axiom*. An *assertion* is a formula $A(c)$ or $P(c, d)$ where $A \in \mathbf{C}$, $P \in \mathbf{R}$, and $c, d \in \mathbf{I}$ (called *positive*) or its negation, i.e., $\neg A(c)$ resp. $\neg P(c, d)$ (*negative*).[1] An example of a $DL\text{-}Lite_{\mathcal{A}}$ ontology is $\mathcal{O}$ in Figure 1. The semantics of DL ontologies $\mathcal{O}$ is based on first-order interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of Σ_o with a domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ [Calvanese *et al.*, 2007] and is captured by a modular translation $\tau(\mathcal{O})$ of $\mathcal{O}$ to first-order logic [Baader *et al.*, 2007]; $\mathcal{O}$ is *consistent*, if $\tau(\mathcal{O})$ is satisfiable. Throughout the paper, we assume that ontologies are in $DL\text{-}Lite_{\mathcal{A}}$, under the unique names assumption as usual.

2.2 Description Logic Programs

DL-rules extend rules in non-monotonic logic programs with special DL-atoms. They are formed over a signature $\Sigma = \langle \mathcal{C}, \mathbf{I}, \mathcal{P}, \mathbf{C}, \mathbf{R} \rangle$ where $\Sigma_o = \langle \mathbf{I}, \mathbf{C}, \mathbf{R} \rangle$ is a DL signature, $\mathcal{C} \supseteq \mathbf{I}$ is a set of constant symbols and $\mathcal{P}$ is a finite set of predicate symbols (called *lp predicates*) of arities ≥ 0 disjoint with $\mathbf{C}, \mathbf{R}$; for simplicity, we assume here $\mathcal{C} = \mathbf{I}$.

Syntax. A *(disjunctive) DL-program* $\Pi = \langle \mathcal{O}, P \rangle$ consists of a DL ontology $\mathcal{O}$ and a finite set P of *DL-rules* r of the form

$$a_1 \vee \ldots \vee a_n \leftarrow b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m\,, \quad (1)$$

where each a_i, $0 \leq i \leq n$ is a first-order atom $p(\vec{t})$ with predicate $p \in \mathcal{P}$ (called *ordinary* or *lp-atom*) and each b_i, $1 \leq i \leq m$, is either an lp-atom or a *DL-atom*; if $n = 0$, the rule is a constraint, and if $n \leq 1$, it is *normal*.

A *DL-atom* $a(\mathbf{t})$ is of form $\mathrm{DL}[\lambda; Q](\vec{t})$, where (a) the list

$$\lambda = S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m, \quad m \geq 0, \quad (2)$$

for each i, $1 \leq i \leq m$, $S_i \in \mathbf{C} \cup \mathbf{R}$, $op_i \in \{\uplus, \cup, \cap\}$ is an *update operator*, and $p_i \in \mathcal{P}$ is an *input predicate* of the same arity as S_i; intuitively, $op_i = \uplus$ (resp., $op_i = \cup$) increases S_i (resp., $\neg S_i$) by the extension of p_i, while $op_i = \cap$ constrains S_i to p_i; (b) $Q(\vec{t})$ is a *DL-query*, which has one of the forms (i) $C(t)$, where C is a concept and t is a term; (ii) $R(t_1, t_2)$, where R is a role and t_1, t_2 are terms; (iii) Q is an inclusion

[1] Negative assertions $\neg F(\vec{t})$ are easily compiled to positive ones using a fresh concept resp. role name $F_{\neg}$ and $F_{\neg}(\vec{t})$, $F_{\neg} \sqsubseteq \neg F$.

axiom and $\vec{t} = \epsilon$; (iv) Q is a disjointness axiom and $\vec{t} = \epsilon$; or (v) $\neg Q'(\vec{t})$ where $Q'(\vec{t})$ is from (i)-(iv). We skip $(\vec{t})$ for $\vec{t} = \epsilon$.

Example 2. In the Example 1 the rule (9) contains a DL-atom $\mathrm{DL}[Male \uplus boy; Male](Y)$, where we first enrich the concept $Male$ in $\mathcal{O}$ by the extension of the predicate boy in P via the operator $\uplus$. We then query the concept $Male$ over the modified version of the ontology.

Semantics. The semantics of a DL-program $\Pi = \langle \mathcal{O}, P \rangle$ is in terms of its grounding $grd(\Pi) = \langle \mathcal{O}, grd(P) \rangle$ over $\mathcal{C}$, i.e., $grd(P)$ contains all ground instances of rules r in P over $\mathcal{C}$. In the remainder, by default we assume Π is ground.

A (Herbrand) *interpretation* of Π is a set $I \subseteq HB_\Pi$ of ground atoms, where HB_Π is the Herbrand base w.r.t. $\mathcal{C}$ and $\mathcal{P}$ (i.e. all ground atoms over $\mathcal{C}$ and $\mathcal{P}$); I satisfies an lp- or DL-atom a, if (i) $a \in I$, if a is an lp-atom, and (ii) $\tau(\langle \mathcal{T}, \mathcal{A} \cup \lambda^I(a) \rangle) \models Q(\mathbf{c})$ where $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, if a is a DL-atom of form (2), where $\lambda^I(a) = \bigcup_{i=1}^m A_i(I)$,

- $A_i(I) = \{S_i(\vec{t}) \mid p_i(\vec{t}) \in I\}$, for $op_i = \uplus$;
- $A_i(I) = \{\neg S_i(\vec{t}) \mid p_i(\vec{t}) \in I\}$, for $op_i = \cup$;
- $A_i(I) = \{\neg S_i(\vec{t}) \mid p_i(\vec{t}) \in HB_\Pi \setminus I\}$, for $op_i = \cap$.

Satisfaction of a DL-rule r resp. set P of rules by I is then as usual, where I satisfies $not\ b_j$ if I does not satisfy b_j; I satisfies Π, if it satisfies each $r \in P$. We denote that I satisfies (is a *model* of) an object o (atom, rule, etc.) with $I \models^{\mathcal{O}} o$.

Example 3. Going back to the Example 1, the interpretation $I = \{ischildof(john, alex), boy(john)\}$ satisfies the DL-atom $o = \mathrm{DL}[Child \uplus boy; \neg Male](john)$, as $\mathcal{O} \cup \lambda^I(o) \models \neg Male(john)$. Furthermore, $I \not\models^{\mathcal{O}} \mathrm{DL}[; Adopted](john)$, since the input list of the respective DL-atom is empty and $\mathcal{O} \not\models Adopted(john)$.

Finally, answer sets of Π are defined using a reduct ρ_x that maps any set P of rules and $I \subseteq HB_\Pi$ to a set $\rho_x P^I$ of rules: I is an *x-answer set* of Π, if I is a $\subseteq$-minimal model of $\rho_x \Pi^I = \langle \mathcal{O}, \rho_x P^I \rangle$ (called the *x-reduct* of Π); $AS_x(\Pi)$ is the set of all x-answer sets of Π. In particular, for weak ($x = weak$) and FLP ($x = \mathit{flp}$) answer sets [Eiter *et al.*, 2005]:

• The $weak$-reduct is $\rho_{weak} P^I = \{wr^I \mid r \in P\}$ where wr^I is void if either $I \models b_j$, for some $k < j \leq m$ or $I \not\models^{\mathcal{O}} b_i$, for some DL-atom b_i, $1 \leq i \leq k$; otherwise, wr^I removes all DL-atoms b_i and all $not\ b_j$ from the rule body.

• The *flp*-reduct is $\rho_{\mathit{flp}} P^I = \{fr^I \mid r \in P\}$ where $fr^I = r$ if the body of r is satisfied, i.e., $I \models^{\mathcal{O}} b_i$, for all b_i, $1 \leq i \leq k$ and $I \not\models^{\mathcal{O}} b_j$, for all $k < j \leq m$; otherwise, fr^I is void.

For further reduct-based answer sets, see e.g. [Eiter *et al.*, 2008b; Lukasiewicz, 2010; Wang *et al.*, 2010; Shen, 2011].

A DL-program is *inconsistent*, if it has no answer set.

Example 4 (cont'd). As mentioned, Π is inconsistent; if we drop (4) from $\mathcal{A}$, then $I = \{ischildof(john, alex), boy(john), contact(john, pat)\}$ is both a weak and an FLP answer set. Along with the facts (7) and (8) the *flp*-reduct $\rho_{\mathit{flp}} P^I$ contains the ground rule (11), where X and Y are substituted by $john$ and pat respectively.

3 Repair Answer Sets

We now turn to repairing an inconsistent DL-program $\langle \mathcal{O}, P \rangle$. In our setting, we assume that the rule part P, which is on top of the ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, is reliable and that the cause for inconsistency is in the latter. Thus when searching for a repair, modifications should only be applied to $\mathcal{O}$. In principle, the TBox $\mathcal{T}$ and the ABox $\mathcal{A}$ of the ontology could be subject to change; however, as usually the TBox is well-developed and suitable TBox change is less clear in general (the more by an external user), we confine to change only the ABox.

Hence given a possibly inconsistent DL-program, our goal is to find an ABox such that replacing the ABox $\mathcal{A}$ by it makes the DL-program consistent. The answer sets of such a "repaired" DL-program are then referred to as repair answer sets of the program. Formally, they are defined as follows.

Definition 5. Given a DL-program $\Pi = \langle \mathcal{O}, P \rangle$, $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$,

- an ABox $\mathcal{A}'$ is an *x-repair* of Π, if (i) $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent, and (ii) $\Pi' = \langle \mathcal{O}', P \rangle$ has some x-answer set; the set of all x-repairs of Π is denoted $rep_x(\Pi)$.
- I is an *x-repair answer set* of Π, if $I \in AS_x(\Pi')$, where $\Pi' = \langle \mathcal{O}', P \rangle$, $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, and $\mathcal{A}' \in rep_x(\Pi)$; the set of all x-repair answer sets of Π is denoted by $RAS_x(\Pi)$.

Furthermore, we denote by $rep_x^I(\Pi) = \{\mathcal{A}' \in rep_x(\Pi) \mid I \in AS_x(\Pi'), \Pi' = \langle \mathcal{O}', P \rangle, \mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle\}$ the set of all ABoxes $\mathcal{A}'$ under which I becomes an x-answer set of Π.

Example 6. Reconsider Π in Example 1. The interpretation $I_1 = \{ischildof(john, alex), boy(john)\}$ is an *flp*-repair answer set with *flp*-repair $\mathcal{A}'_1 = \{Male(pat), Male(john)\}$.

We first briefly address the complexity of repair answer sets, but refrain from an extensive complexity study here.

Theorem 7. *Given a ground DL-program Π, deciding whether $RAS_x(\Pi) \neq \emptyset$ is (a) Σ_2^P-complete for arbitrary Π and $x \in \{\mathit{flp}, weak\}$; (b) Σ_2^P-complete for normal Π and $x = \mathit{flp}$; (c) NP-complete for normal Π and $x = weak$. Deciding $AS_x(\Pi) \neq \emptyset$ has in all cases the same complexity as deciding $RAS_x(\Pi) \neq \emptyset$.*

Nonsurprisingly, repair answer sets and ordinary answer sets have the same complexity in general. Indeed, a repair $\mathcal{A}'$ can be jointly guessed with an answer set I, and the test whether I is a minimal model of $\rho_x \Pi'^I$ can be done as usual; the upper bounds are then immediate from the following:

Proposition 8. *Given any $I \subseteq HB_\Pi$, $\mathcal{O}$ in DL-Lite$_\mathcal{A}$, and DL-atom a, deciding $I \models^{\mathcal{O}} a$ is feasible in polynomial time.*

This proposition is straightforward from the definition of the DL-atom and a well-known result presented in [Calvanese *et al.*, 2007]. The hardness is inherited from ordinary answer set programs, except for normal FLP answer sets, for which a common Σ_2^p-hardness proof of ordinary disjunctive logic programs can be adapted.

3.1 Selection Preference

Clearly, not all repairs are equally useful or interesting for a certain scenario. For instance, repairs that have no common assertions with the original ABox might be unwanted; repairs that introduce assertions that are not in the initial ABox; repairs that would cause non-minimal change etc.

Formally, we model preferred repairs using a *selection* function $\sigma : 2^{\mathcal{AB}} \times \mathcal{AB} \to 2^{\mathcal{AB}}$, where $\mathcal{AB}$ is the set of all ABoxes, that given a set S of ABoxes and an ABox $\mathcal{A}$, returns a set $\sigma(S, \mathcal{A}) \subseteq S$ of *preferred* (or *selected*) ABoxes.

This notion captures a variety of selection principles, including minimal repairs according to some preference relation, or some global selection property. We then define

Definition 9. Given $\Pi = \langle \mathcal{O}, P \rangle$, $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, and a selection σ, we call $rep_{(\sigma,x)}(\Pi) = \sigma(rep_x(\Pi), \mathcal{A})$ the (σ, x)-*repairs* of Π. An interpretation $I \subseteq HB_\Pi$ is a (σ, x)-*repair answer set* of Π, if $rep^I_{(\sigma,x)}(\Pi) \neq \emptyset$, where $rep^I_{(\sigma,x)}(\Pi) = rep_{(\sigma,x)}(\Pi) \cap rep^I_x(\Pi)$; $RAS_{(\sigma,x)}(\Pi)$ is the set of all such answer sets.

In general, even polynomially computable selections σ may incur intractability, like e.g. selecting ABoxes $\mathcal{A}'$ with set-minimal change to $\mathcal{A}$, or with smallest Dalal (Hamming) distance. We aim at selections that are useful in practice and have benign computational properties. Useful in this regard are selections of the following kind.

Definition 10. A selection $\sigma : 2^{\mathcal{AB}} \times \mathcal{AB} \to 2^{\mathcal{AB}}$ is *independent*, if $\sigma(S, \mathcal{A}) = \sigma(S', \mathcal{A}) \cup \sigma(S \setminus S', \mathcal{A})$ whenever $S' \subseteq S$.

Independence allows us to decide whether a given repair $\mathcal{A}' \in S$ is selected by σ without looking at other repairs, and composition works easily. This makes the introduced property valuable, since independent selection functions of different kind can be conveniently combined without a major increase in the complexity. Clearly, set-minimal change and smallest Dalal distance are not independent. On the other hand, selecting all ABoxes such that $\mathcal{A}' \subseteq \mathcal{A}$, is obviously independent. The latter, and several other independent selections that are useful in practice, will be considered in Section 5.

Independence leads to the following beneficial property.

Proposition 11. *For every* Π *and selection* σ*, if* σ *is independent, then* $rep^I_{(\sigma,x)}(\Pi) \subseteq rep_{(\sigma,x)}(\Pi)$*, for every* $I \subseteq HB_\Pi$*.*

Proof (Sketch). By definition $rep_{(\sigma,x)}(\Pi) = \sigma(rep_x(\Pi), \mathcal{A})$ and $rep^I_{(\sigma,x)}(\Pi) = \sigma(rep^I_x(\Pi), \mathcal{A})$. Now since $rep^I_x(\Pi) \subseteq rep_x(\Pi)$ and σ is independent, we obtain $\sigma(rep_x(\Pi), \mathcal{A}) = \sigma(rep^I_x(\Pi), \mathcal{A}) \cup \sigma(rep_x(\Pi) \setminus rep^I_x(\Pi), \mathcal{A})$, from which the result follows. □

Proposition 11 implies that if we can turn an interpretation I into an answer set of Π by a σ-selected repair from the repairs which achieve this for I, then I is a σ-repair answer set of Π; that is, *local selection* is enough for a *global* σ-repair answer set. This will be exploited in the next section.

4 Computation

In this section we first recall the essentials of the evaluation algorithm for DL-programs in [Eiter *et al.*, 2012] (given there for the more general class of so-called HEX-programs). We then present the core procedure of an extension for computing (σ, x)-repair answer sets.

4.1 Evaluation of DL-programs

DL-program evaluation builds on a rewriting $\hat{\Pi}$ of Π, where DL-atoms a are replaced by ordinary atoms (replacement atoms) e_a, together with a guess on their truth by additional 'choice' rules. Given an interpretation $\hat{I}$ over this extended language, we use $\hat{I}|_\Pi$ to denote its restriction to the original language of Π. A crucial notion is that of compatibily:

Algorithm 1: $AnsSet$: Compute $AS_x(\Pi)$

Input: A DL-program Π, $x \in \{weak, \mathit{flp}\}$
Output: $AS_x(\Pi)$
for $\hat{I} \in AS(\hat{\Pi})$ **do**
(a) **if** $CMP(\hat{I}, \Pi) \wedge xFND(\hat{I}, \Pi)$ **then**
output $\hat{I}|_\Pi$
end
end

Definition 12. [Eiter *et al.*, 2012] A *compatible set* of a DL-program $\Pi = \langle \mathcal{O}, P \rangle$ is an interpretation $\hat{I}$, such that

(i) $\hat{I}$ is an answer set of $\hat{\Pi}$, and
(ii) $e_a \in \hat{I}$ iff $\hat{I}|_\Pi \models^{\mathcal{O}} a$, for all $a = \mathrm{DL}[\lambda; Q](c)$ of Π.

Conversely, given an interpretation I of Π, we denote by I_c the interpretation of $\hat{\Pi}$, such that I_c coincides with I on nonreplacement atoms, and each replacement atom e_a is in I_c (i.e., true) iff $I \models^{\mathcal{O}} a$ for the respective DL-atoms a.

With these concepts in place, we are ready to describe the basic algorithm $AnsSet$ (cf. Algorithm 1) for evaluating a DL-program $\Pi = \langle \mathcal{O}, P \rangle$ adapted from [Eiter *et al.*, 2012].

First, $\hat{\Pi}$ is evaluated by an ordinary ASP solver. For every answer set $\hat{I}$, in (a), the function CMP checks for compatibility, while $xFND$ tests for foundedness, i.e., whether $\hat{I}|_\Pi$ is a $\subseteq$-minimal model of the reduct $\rho_x \Pi^{\hat{I}|_\Pi}$. In case of $x = weak$ it just returns true, otherwise ($x = \mathit{flp}$) it checks for disjointness with unfounded sets as defined in [Eiter *et al.*, 2012]). If both tests succed, then $\hat{I}|_\Pi$ is output as an answer set.

An important link between the answer sets of Π and $\hat{\Pi}$ is:

Proposition 13. *If* $I \in AS_x(\Pi)$ *then* $I_c \in AS_x(\hat{\Pi})$*.*

While $AnsSet$ is clearly sound, from this result its completeness follows, i.e. restricting the search to $AS_x(\hat{\Pi})$ does not yield any loss of answer sets.

4.2 Repair Computation

In this subsection, we first aim at a procedure for computing (σ, x)-repairs given an independent selection function σ. Then, we briefly describe how its main subroutine can be used for an extension of $AnsSet$ that computes answer sets if they exist, and (σ, x)-repair answer sets otherwise.

A first key observation is that Proposition 13 can be generalized to repair answer sets, more precisely:

Proposition 14. *If* $I \in RAS_x(\Pi)$ *then* $I_c \in AS(\hat{\Pi})$*.*

Proof (Sketch). By definition of $RAS_x(\Pi)$, we get that $I \in AS(\Pi')$, where $\Pi' = \langle \mathcal{O}', P \rangle$, $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ and $\mathcal{A}' \in rep_x(\Pi)$. Since by Proposition 13 $I_c \in AS(\hat{\Pi}')$ and $\hat{\Pi} = \hat{\Pi}'$, the result immediately follows. □

Thus, our approach is to traverse $AS(\hat{\Pi})$ and check for each answer set $\hat{I}$ whether it is a (σ, x)-repair answer set of Π. The latter proceeds in two steps, where the first step is to search for potential σ-repairs of the ontology such that Definition 12(ii) holds for $\hat{I}$. More formally (and slightly generalized), we define an ontology repair problem as follows.

Algorithm 2: $RepAns$: Compute $rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$

Input: $\Pi{=}\langle\mathcal{O},P\rangle$, $\mathcal{O}{=}\langle\mathcal{T},\mathcal{A}\rangle$, $\hat{I}{\in}AS(\hat{\Pi})$, σ, $x{\in}\{weak, flp\}$
Output: $rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$
for $\mathcal{A}' \in ORP(\hat{I},\Pi,\sigma)$ **do**
  if $CMP(\hat{I},\langle\mathcal{T},\mathcal{A}',P\rangle) \wedge xFND(\hat{I},\langle\mathcal{T},\mathcal{A}',P\rangle)$ **then**
    output $\mathcal{A}'$
  end
end

Definition 15. An ontology repair problem (ORP) is a triple $\mathcal{P} = \langle\mathcal{O}, D_1, D_2\rangle$ where $\mathcal{O} = \langle\mathcal{T},\mathcal{A}\rangle$ is an ontology and $D_i = \{\langle U^i_j, Q^i_j\rangle \mid 1 \leq j \leq m_i\}$, $i = 1, 2$, are sets of pairs where each U^i_j is an ABox and each Q^i_j is a DL-query. A *repair* (*solution*) for $\mathcal{P}$ is any ABox $\mathcal{A}'$ such that

(i) the ontology $\mathcal{O}' = \langle\mathcal{T},\mathcal{A}'\rangle$ is consistent;
(ii) $\tau(\langle\mathcal{T},\mathcal{A}' \cup U^1_j\rangle) \models Q^1_j$ holds for $1 \leq j \leq m_1$;
(iii) $\tau(\langle\mathcal{T},\mathcal{A}' \cup U^2_j\rangle) \not\models Q^2_j$ holds for $1 \leq j \leq m_2$.

For a selection σ, the *σ-repairs* of $\mathcal{P}$ are σ-$rep(\mathcal{P}) = \sigma(rep(\mathcal{P}),\mathcal{A})$, where $rep(\mathcal{P})$ is the set of all repairs of $\mathcal{P}$.

Intuitively every pair $\langle U^i_j, Q^i_j\rangle$ corresponds to some DL-atom $a^i_j = \mathrm{DL}[\lambda^i_j; Q^i_j](\vec{t})$ under an interpretation I, i.e., such that $U^i_j = \lambda^I(a^i_j)$. Moreover, a^1_j (resp., a^2_j) should evaluate to true (resp. false) such that I is a (potential) answer set of Π. We illustrate these ideas by an example, which also shows how an answer set $\hat{I}$ of $\hat{\Pi}$ induces a corresponding ORP instance.

Example 16. Let $\Pi = \langle\mathcal{O}, P\rangle$ be a DL-program, where

$$P = \left\{ \begin{array}{l} p(c);\ \ r(c);\ \ q(c) \leftarrow \mathrm{DL}[C{\uplus}r; D](c); \\ \bot \leftarrow\ \ \mathrm{DL}[D \uplus p, E{\uplus}r; \neg C](c) \end{array} \right\}.$$

Let $a_1 = \mathrm{DL}[C{\uplus}r; D](c)$ and $a_2 = \mathrm{DL}[D \uplus p, E{\uplus}r; \neg C](c)$, and consider the interpretation $\hat{I} = \{p(c), r(c), q(c),\ e_{a_1}\}$, i.e., a_1 is guessed true and a_2 false. The corresponding ORP is given by $\mathcal{P} = \langle\mathcal{O}, D_1, D_2\rangle$, where $D_1 = \{\langle\{\neg C(c)\}; D(c)\rangle\}$ and $D_2 = \{\langle\{D(c), \neg E(c)\}; \neg C(c)\rangle\}$.

The procedure $RepAns$ (cf. Algorithm 2) calls the subroutine $ORP(\hat{I},\Pi,\sigma)$ to compute σ-repairs of the corresponding ORP. Further on, $RepAns$ re-uses functions CMP and $xFND$ to check whether $\hat{I}$ is an answer set of $\hat{\Pi}'$ and that it is founded w.r.t. $\Pi' = \langle\mathcal{O}', P\rangle$, $\mathcal{O}' = \langle\mathcal{T},\mathcal{A}'\rangle$. It thus computes the set of all ABoxes under which $\hat{I}$ becomes a (σ, x)-repair answer set.

Let then $RepAnsSet$ be the algorithm that iteratively calls $RepAns$ for every $\hat{I} \in AS(\hat{\Pi})$, and that outputs any $\hat{I}$ where the result is nonempty. We then can show:

Theorem 17. *$RepAns$ and $RepAnsSet$ are sound and complete for $rep_{(\sigma,x)}(\Pi)$ and $RAS_{(\sigma,x)}(\Pi)$, respectively, for independent selection σ.*

Proof (Sketch). Soundess. Let $\mathcal{A}'$ be an output of $RepAns$. Towards contradiction, suppose $\mathcal{A}' \notin rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$. Then $\hat{I}|_\Pi \notin AS(\Pi')$, where $\Pi' = \langle\mathcal{T},\mathcal{A}',P\rangle$ and $\mathcal{A}'$ is σ-selected. Clearly $\mathcal{A}'$ is σ-selected, since otherwise $\mathcal{A}' \notin ORP(\hat{I},\Pi,\sigma)$ and $\mathcal{A}'$ is not in the output. As it is given that $\hat{I} \in AS(\hat{\Pi})$, it must hold that either $\hat{I}$ is not a compatible set of Π' or it is not x-founded. If either of these is true then the corresponding procedure CMP or $xFND$ returns false and $\mathcal{A}'$ is not in the output, which leads to contradition.

Completeness. Let $rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$ be the set of all σ-selected repairs for Π that turn $\hat{I}|_\Pi$ into an x-repair answer set. Towards contradiction assume that there exists some $\mathcal{A}' \in rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$, which is not an output of the algorithm $RepAns$. Then either (1) $\mathcal{A}' \notin ORP(\hat{I},\Pi,\sigma)$; (2) $CMP(\hat{I},\langle\mathcal{T},\mathcal{A}',P\rangle) = false$ or (3) $xFND(\hat{I},\langle\mathcal{T},\mathcal{A}',P\rangle) = false$. If (1) holds, then $\mathcal{A}'$ is not a solution of the ORP problem. Thus either $\langle\mathcal{T},\mathcal{A}'\rangle$ is unsatisfiable (contradiction to $\mathcal{A}' \in rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$ by the definition of the repair) or the actual values of the DL-atoms do not coincide with the replacement atoms in $\hat{\Pi}$ (contradiction due to the failure of the compatibility check). Finally, if either (2) or (3) holds then we obtain contradiction, since $\mathcal{A}' \in rep^{\hat{I}|_\Pi}_{(\sigma,x)}(\Pi)$ implies $\hat{I}|_\Pi$ should be compatible and x-founded.

Using argument above and Proposition 14 soundness and completeness of $RepAnsSet$ is easily established. □

Similarly, $RepAns$ could be intertwined with $AnsSet$ for an extension that computes answer sets if they exist, and repair answer sets otherwise: while iterating over $\hat{I} \in AS(\hat{\Pi})$ and checking for compatibility and foundedness, also $RepAns$ is called for every $\hat{I}$, as long as no answer set is found. The results of these calls are stored and can, in case, eventually be used to output all (σ, x)-repair answer sets.

Note also that, for illustration, we kept the algorithms simple and several optimizations apply. For instance, to compute some (σ, x)-repair answer set, we can replace $RepAns$ by a version that just computes a first witnessing ABox $\mathcal{A}'$. Moreover, caching ABoxes $\mathcal{A}'$ and/or all answer sets of the respective Π' (which can be straight output as (σ, x)-repair answer sets of Π) further reduces the search space.

A natural question is whether computing repair answer sets via compatible sets $\hat{I}$ of Π makes repair answer set checking for $\hat{I}|_\Pi$ easier than for abritrary interpretations I. Unfortunately, this is not the case.

Theorem 18. *For ground $\Pi = \langle\mathcal{O}, P\rangle$ and $I \subseteq HB_\Pi$, deciding whether $I{\in}RAS_x(\Pi)$ is* NP*-complete for $x{=}weak$ and Σ^p_2-complete for $x{=}flp$; hardness holds even if $I = \hat{I}|_\Pi$ for a compatible set $\hat{I}$ of Π.*

Proof (Sketch). For space issues the exposition is necessarily superficial and covers only the Σ^p_2-hardness result. The latter holds by a reduction from the problem of validity of a QBF formula

$$\phi = \exists x_1 \ldots x_n \forall y_1 \ldots y_m E, \quad n, m \geq 1, \tag{3}$$

where $E = \chi_1 \vee \ldots \vee \chi_r$ is a DNF formula, and each $\chi_k = l_{k_1} \wedge l_{k_2} \wedge l_{k_3}$ is a conjunction of literals over atoms $x_1, \ldots, x_n, y_1, \ldots, y_m$. Given ϕ, we construct $\Pi = \langle\emptyset,\mathcal{A},P\rangle$ with $\mathcal{A} = \{X_1(a), \ldots, X_n(a)\}$ and P as follows:

$$P = \left\{ \begin{array}{l} (1)\ \bot \leftarrow \mathit{not}\ \mathrm{DL}[;X_i](a), \mathit{not}\ \mathrm{DL}[;\neg X_i](a). \\ (2)\ \bot \leftarrow \mathrm{DL}[;Y_j](a). \\ (3)\ \bot \leftarrow \mathrm{DL}[;\neg Y_j](a). \\ (4)\ w(a) \leftarrow \mathit{not}\ w(a). \\ (5)\ y_j(a) \leftarrow w(a). \\ (6)\ w(a) \leftarrow f(l_{k_1}), f(l_{k_2}), f(l_{k_3}). \end{array} \right\},$$

where $f(x_i) = \mathrm{DL}[X_i \uplus w; X_i](a)$,
$f(\neg x_i) = \mathrm{DL}[X_i \cup w; \neg X_i](a)$,
$f(y_j) = \mathrm{DL}[Y_j \uplus y_j, Y_j \uplus w; Y_j](a)$,
$f(\neg y_j) = \mathrm{DL}[Y_j \cap y_j, Y_j \cup w; \neg Y_j](a)$.

Set $I = \{w(a), y_1(a), \ldots, y_m(a)\}$, and note that $I = \hat{I}|_\Pi$ for some compatible set $\hat{I}$ of Π and $f\Pi^I = \{(5), (6)\}$. Intuitively, due to (1)-(3) a repair $\mathcal{A}'$ must be a maximal consistent subset of $\{X_i(a), \neg X_i(a) \mid 1 \leq i \leq n\}$. Now $I \in RAS_{flp}(\Pi)$ implies the existence of some $\mathcal{A}'$ s.t. by minimality of I, for each $I' \subseteq I \backslash \{w(a)\}$ exists some k, where all $f(l_{k_1}), f(l_{k_2}), f(l_{k_3})$ are true, hence χ_k is true; thus ϕ is true. Conversely, every assignment to the x_i witnessing that ϕ is true induces some $\mathcal{A}' \in rep^I_{flp}(\Pi)$. □

This also yields a lower bound for (σ, x)-repair answer sets. Informally, the reason is that ORPs are a source of complexity that is orthogonal to minimality checking of models.

An interesting and practically relevant quest is for tractable cases, and a natural choice is to consider restricted programs and ORPs (see next section), such that compatibility checking is polynomial. While this yields tractability for weak-repair answer sets, checking foundedness for *flp*-repair answer sets remains intractable (although the complexity drops to NP).

5 Ontology Repair

The Ontology Repair Problem from above is an important subtask in the algorithm for computing repair answer sets. Nonsurprisingly, this problem is intractable in general; however, this holds already in simple settings.

Theorem 19. *Deciding if an ORP $\mathcal{P}=\langle \mathcal{O}, D_1, D_2\rangle$ has some repair is* NP*-complete, and* NP*-hard even if $\mathcal{T}=\mathcal{A}=\emptyset$.*

Proof (Sketch). A guess for a repair $\mathcal{A}'$ is verifiable in polynomial time, as deciding each $\langle \mathcal{T}, \mathcal{A}' \cup U^i_j\rangle \models Q^i_j$ is polynomial. NP-hardness holds by a reduction from SAT. Given $\phi = \chi_1 \wedge \cdots \wedge \chi_m$ on atoms $x_1, \ldots, x_n$, we construct $\mathcal{P} = \langle\langle \emptyset, \emptyset\rangle, D_1, D_2\rangle$, with concepts $X_j, \bar{X}_j$ for the x_j and A, s.t.

- $D_1 = \{\langle U_i, A(d)\rangle, \langle V_j, A(d)\rangle \mid 1 \leq i \leq m\}$, where $U_i = \{\bar{X}_j(d), X_{j'}(d) \mid x_j \in \chi_i, \neg x_{j'} \in \chi_i\}$, $V_j = \{X_j(d), \bar{X}_j(d)\}$,
- $D_2 = \{\langle \emptyset, A(d)\rangle\}$.

Intuitively, by D_2 a repair $\mathcal{A}'$ must not contain $A(d)$, and by D_1 adding either (i) U_i or (ii) V_j to $\mathcal{A}'$ causes inconsistency. By (i) $\mathcal{A}'$ must contain some $X_j(d)$ (resp. $\bar{X}_j(d)$) such that $x_j \in \chi_i$ ($\neg x_j \in \chi_i$), and by (ii) $\{X_j(d), \bar{X}_j(d)\} \not\subseteq \mathcal{A}'$; thus $\mathcal{A}'$ encodes a consistent choice of literals that satisfies ϕ. □

We note that ORP has two sources of NP-hardness, viz. the data part (as in the proof above), and the taxonomy, which under σ-repairs may derive further assertions. Furthermore, each ORP can be encountered in the algorithm above, for some answer set $\hat{I}$ of a program $\hat{\Pi}$; we show this on an example.

Example 20. Let $\mathcal{P} = \langle \mathcal{O}, D_1, D_2\rangle$ where $D_1 = \{\delta_1\}$, $D_2 = \{\delta_2\}$ with $\delta_1 = \langle\{C(c), \neg D(c)\}, \neg E(c)\rangle$, and $\delta_2 = \langle\{D(d), \neg S(d)\}, C(d)\rangle$. We introduce predicates $p^{\delta_1}_C, p^{\delta_1}_D$ for δ_1 and $p^{\delta_2}_D, p^{\delta_2}_S$ for δ_2 and construct $\Pi = \langle \mathcal{O}, P_I \cup P_{DL}\rangle$, where

$$P_I = \{p^{\delta_1}_C(c);\ p^{\delta_1}_D(c);\ p^{\delta_2}_D(d);\ p^{\delta_2}_S(d)\},$$

$$P_{DL} = \left\{ \begin{array}{ll} \bot \leftarrow \mathit{not}\ \mathrm{DL}[C \uplus p^{\delta_1}_C, D \uplus p^{\delta_1}_D; \neg E](c); & (1) \\ \bot \leftarrow \mathrm{DL}[D \uplus p^{\delta_2}_D,\ S \uplus p^{\delta_2}_S; C](d)\} & (2) \end{array} \right\}.$$

Then, $\hat{\Pi}$ has a single answer set $\hat{I}$, and it gives rise to $\mathcal{P}$; (1) effects the pair δ_1 in D_1 and (2) the pair δ_2 in D_2.

5.1 Tractable Cases

To gain tractability for ORP, we consider restrictions on repairs and the ontology, as the pairs D_1 and D_2 are hard to control in practice. We present four tractable cases of σ-repairs with independent selection function σ, which are arguably useful in practice. In what follows, let $\mathcal{P} = \langle \mathcal{O}, D_1, D_2\rangle$, $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$.

1. Bounded $\delta^{\pm}$-change. A natural restriction is to bound the distance from the original ABox, i.e., use $\sigma_{\delta^{\pm},k}(S, \mathcal{A}) = \{\mathcal{A}' \mid |\mathcal{A}' \triangle \mathcal{A}| \leq k\}$, for some constant k. As the number m of possible ABox assertions is polynomial in the size of $\mathcal{T}$ and $\mathcal{A}$, traversing all $O(\binom{m}{k})$ possible $\mathcal{A}'$ and checking the repair condition is possible in polynomial time.

2. Deletion repair. Another important restriction is to allow only to delete assertions from the original ABox, i.e., use $\sigma_{del}(S, \mathcal{A}) = \{\mathcal{A}' \mid \mathcal{A}' \subseteq \mathcal{A}\}$.

Example 21. In Example 1, each $\mathcal{A}' \subset \mathcal{A}$ except $\{Male(pat), hasParent(john, pat)\}$ is a deletion repair.

To achieve tractability, we exclude non-containment ($\not\sqsubseteq$) DL-queries, i.e., of the form $\neg Q$ where Q is an inclusion or disjointness axiom, from $\mathcal{P}$; let us call such ORPs $\not\sqsubseteq$*-free*.

Theorem 22. *Deciding if a $\not\sqsubseteq$-free ORP $\mathcal{P} = \langle \mathcal{O}, D_1, D_2\rangle$ with consistent $\mathcal{O}$ has a σ_{del}-repair is polynomial.*

For inconsistent $\mathcal{O}$, or if non-containment DL-queries occur in $\mathcal{P}$, the problem is NP-hard. The proof of Proposition 22 exploits the following property of the remaining DL-queries.

Lemma 23. *If $\langle \mathcal{T}, \mathcal{A}\rangle$ is consistent, then $\tau(\langle \mathcal{T}, \mathcal{A} \cup U^i_j\rangle) \models Q^i_j$ iff $\tau(\langle \mathcal{T}, \mathcal{A}_0 \cup U^i_j\rangle) \models Q^i_j$ for some $\mathcal{A}_0 \subseteq \mathcal{A}$ with $|\mathcal{A}_0| \leq 1$.*

That is, at most one assertion α from $\mathcal{A}$ is sufficient to derive the query. This follows from a respective result for empty U^i_j and instance queries Q^i_j in [Calvanese *et al.*, 2007].

Now if $\tau(\langle \mathcal{T}, U^i_j\rangle) \models Q^i_j$, we can drop $\langle U^i_j, Q^i_j\rangle$ from $\mathcal{P}$ if $i=1$, and stop if $i=2$ as no repair exists. Otherwise we let the *support set* $Supp^i_j$ of Q^i_j contain all assertions α such that $\tau(\langle \mathcal{T}, \{\alpha\} \cup U^i_j\rangle) \models Q^i_j$. Then, any repair $\mathcal{A}'$ must fulfill $\mathcal{A}' \cap Supp^1_j \neq \emptyset$ for each j (i.e., be a *hitting set*), and must be disjoint with each $Supp^2_{j'}$. Let then $\mathcal{S}_j := (Supp^1_j \cap \mathcal{A}) \setminus \bigcup_{j'} Supp^2_{j'}$. A σ_{del}-repair $\mathcal{A}'$ exists iff each $\mathcal{S}_j$ is nonempty; the hitting sets of the $\mathcal{S}_j$ are all the σ_{del}-repairs. The construction of the $\mathcal{S}_j$ and the check can be done in polynomial time, thus the overall problem is tractable. Furthermore, the (possibly exponentially many) σ_{del}-repairs can be output in

total polynomial time. This method can be extended to allow in addition also a constant number of new assertions in repairs.

3. Deletion δ^+. This selection combines deletion and small change in a prioritized way. First one deletes assertions from $\mathcal{A}$ (assumed to be consistent) according to some polynomial method μ (using domain knowledge etc.) until some $\mathcal{A}_0 = \mu(\mathcal{O}) \subseteq \mathcal{A}$ results that satisfies Definition 15 (iii). If $\mathcal{A}_0$ is a repair, it is the result; otherwise, one looks for a close repair with bounded $\delta^{\pm}$ change. That is $\sigma_{del,\delta^+}(S,\mathcal{A}) = \{\mu(\mathcal{O})\}$ if $\mu(\mathcal{O}) \in S$ and $\sigma_{del,\delta^+}(S,\mathcal{A}) = \sigma_{\delta^+}(S,\mu(\mathcal{O}))$ otherwise.

Example 24 (cont'd). If $\mu(\mathcal{O})$ drops unreliable information about the sex of certain persons (e.g. pat), $\mathcal{A}_0$=$\{Male(john), hasParent(john, pat)\}$ is a deletion repair. If the constraint

$$\bot \leftarrow \mathrm{DL}[;hasParent](X,Y), \\ not\ \mathrm{DL}[;Male](Y), not\ \mathrm{DL}[;Female](Y)$$

(the sex of parents must be known) were in P, one would have to add $Female(pat)$ to $\mathcal{A}_0$ to obtain a deletion-δ^+ repair.

4. Addition under bounded opposite polarity. Repairs by unbounded additions become tractable, if few of them are positive resp. negative, i.e., the number of assertions with opposite polarity is bounded (which by Proposition 19 is necessary). That is, if $\mathcal{A}^+$ (resp., $\mathcal{A}^-$) is the positive (negative) part of an ABox $\mathcal{A}$, then $\sigma_{bop}(S,\mathcal{A}) = \{\mathcal{A}' \supseteq \mu(\mathcal{O}) \mid |{\mathcal{A}'}^+ \backslash \mathcal{A}| \leq k$ or $|{\mathcal{A}'}^- \backslash \mathcal{O}| \leq k\}$ where $\mu(\mathcal{O})$ is a starting ABox as above and k is a constant. The following result is instrumental.

Theorem 25. *For a $\not\sqsubseteq$-free ORP $\mathcal{P} = \langle \mathcal{O}, D_1, D_2\rangle$, $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$, where $\mathcal{T}$ has no disjointness axioms[2] and $\mu(\mathcal{O}) = \mathcal{A}$, deciding whether some σ_{bop}-repair exists is polynomial.*

This can be shown by an extension of the method for deletion repairs; like for the latter, the problem is NP-hard in presence of $\not\sqsubseteq$ DL-queries. Assuming that $\langle \mathcal{T}, \mathcal{A}\rangle$ is consistent (otherwise no σ_{bop}-repair exists), we proceed as follows:

1. Like for deletion repairs, we compute the support sets $Supp^i_j$ and simplify $\mathcal{P}$ resp. quit if no repair can exist, checking also whether $Supp^i_j \cap \mathcal{A} \neq \emptyset$ (as then Q^i_j is entailed).

2. We then let $\mathcal{S}_j = Supp^1_j \setminus (\mathcal{A} \cup \bigcup_{j'} Supp^2_{j'})$. Similar as above, the σ_{bop} repairs are then of the form $\mathcal{A}' = \mathcal{A} \cup \mathcal{H}$ where $\mathcal{H}$ is a hitting set of the $\mathcal{S}_j$, but we must ensure that $\langle \mathcal{T}, \mathcal{A}'\rangle$ is consistent as $\mathcal{H}$ consists of new assertions.

3. We choose a set $\mathcal{H}^- \subseteq \bigcup_j \mathcal{S}_j$ of at most k negative assertions, which is a partial hitting set, and check that $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{H}^-\rangle$ is consistent. If yes, we remove $\mathcal{S}_j$ if it intersects with $\mathcal{H}^-$ and remove otherwise from $\mathcal{S}_j$ each positive assertion α s.t. $\neg\alpha$ is entailed by $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{H}^-\rangle$, and all negative assertions.

4. Then, for every hitting set $\mathcal{H}^+$ of the remaining updated $\mathcal{S}_j$, the ABox $\mathcal{A}' = \mathcal{A} \cup \mathcal{H}^- \cup \mathcal{H}^+$ is a σ_{bop}-repair. On the other hand, some σ_{bop}-repair with few negative additions exists only if some choice for $\mathcal{H}^-$ succeeds.

The crucial point for the correctness of this method is that, if $\mathcal{T}$ has no disjointness axioms, by adding to $\mathcal{A} \cup \mathcal{H}^-$ positive assertions $\mathcal{H}^+$ we can not infer new negative assertions, unless inconsistency emerges; this is exploited in Step 3, which limits the candidate space for positive hitting sets *a priori*. The case of few positive additions is completely symmetric.

[2]Disregarding axioms $F_\neg \sqsubseteq \neg F$ to compile negative assertions.

5. Applicability of results. Like for relational databases, our tractable cases fit real applications, e.g. in case of deletion repairs (observing that non-subsumption queries are insignificant for practical DL-programs) and scenarios akin to key-constraint violations in databases.

Composability of independent selections adds to their applicability. Moreover, they may be combined with DB-style factorization and localization techniques (see [Bertossi, 2011] and references therein) and with local search (see Conclusions) to compute closest repairs.

Bounding the number of changes, especially additions, is also compliant with practice, where too many potential repairs suggest human intervention (cf. database repair).

Finally, one may increase the bound in iterative deepening (assuming that not many changes are needed).

6 Related Work and Conclusions

Managing inconsistent DL-programs has focused so far on inconsistency tolerance, rather than on repair as we considered. Pührer *et al.* [2010] avoid unintuitive answer sets caused by inconsistency in DL-atoms, and dynamically deactivate rules to discard spoiled information; they pointed ontology repair out as an issue, but left it open. Fink [2012] presented a paraconsistent semantics, based on the logic of here-and-there.

Repairing ontologies was considered in many works, often to deal with inconsistency. In particular, [Lembo *et al.*, 2010; Bienvenu, 2012] studied consistent query answering over DL-Lite ontologies based on the repair technique (see [Bertossi, 2011]), using minimal deletion repairs (which amount to non-independent σ-selections). Calvanese *et al.* [2012] studied query answers to $DL\text{-}Lite_{\mathcal{A}}$ ontologies that miss expected tuples, and defined abductive explanations corresponding to repairs. They analyzed the complexity of explanation existence for various preferences that amount to non-independent σ-selections. While their problem can be viewed as a special ORP for atomic queries, we deal–also in comparison to the aforementioned works–with a more general problem, where mixed entailment and non-entailment of queries must be satisfied, and moreover under ABox updates. Repairing inconsistent nonmonotonic logic programs is less developed; [Sakama and Inoue, 2003] used extended abduction to delete minimal sets of rules (but also adding rules can remove inconsistency).

Future Work. There are several issues for ongoing and future work. One is extending this work to other classes of Description Logics, like the tractable DLs $\mathcal{EL}$++ and RL, but also to more expressive DLs like Horn-$\mathcal{SHIQ}$, $\mathcal{SHIQ}$, or $\mathcal{SROIQ}$. Related to this is also to consider DL-programs with richer queries, e.g., with (unions of) conjunctive queries (cf. [Eiter *et al.*, 2008a]), or more generally to consider logic programs that access multiple and perhaps also heterogenous ontologies. Orthogonal to this are further σ-selections for repairs, both independent and non-independent ones. The latter may cause intractability in very simple settings, as they open an exponential search space (e.g., subset-minimal or Dalal distance). However, neighborhood search on top of σ allows to compute locally optimal σ-repairs without loss of tractability (by using a slight adaption of our algorithm). Finally, optimization and implementation remain to be done.

References

[Baader *et al.*, 2007] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, 2nd edition, 2007.

[Bertossi, 2011] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, Ottawa, Canada, 2011.

[Bienvenu, 2012] Meghyn Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 705–711, Toronto, Ontario, Canada, July 2012. American Association for Artificial Intelligence.

[Calvanese *et al.*, 2007] Diego Calvanese, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, October 2007.

[Calvanese *et al.*, 2012] Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni. The complexity of explaining negative query answers in DL-Lite. In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning*, Rome, Italy, June 2012. American Association for Artificial Intelligence.

[Eiter *et al.*, 2005] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 90–96. Professional Book Center, 2005.

[Eiter *et al.*, 2008a] Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, and Roman Schindlauer. Exploiting conjunctive queries in description logic programs. *Annals of Mathematics and Artificial Intelligence*, 53(1–4):115–152, August 2008.

[Eiter *et al.*, 2008b] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12-13):1495–1539, August 2008.

[Eiter *et al.*, 2012] Thomas Eiter, Michael Fink, Thomas Krennwallner, Christoph Redl, and Peter Schüller. Exploiting unfounded sets for HEX-program evaluation. In *Proceedings of 13th European Conference on Logics in Artificial Intelligence*, pages 160–175, Toulouse, France, September 2012. Springer.

[Fink, 2012] Michael Fink. Paraconsistent hybrid theories. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 13th International Conference, KR*, pages 141–151, Rome, Italy, June 2012. American Association for Artificial Intelligence.

[Lembo *et al.*, 2010] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. Inconsistency-tolerant semantics for description logic ontologies. In *Proceedings of the 19th Italian Symposium on Advanced Database Systems*, pages 103–117, Bressanone/Brixen, Italy, September 2010. Springer.

[Lukasiewicz, 2010] Thomas Lukasiewicz. A novel combination of answer set programming with description logics for the semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1577–1592, November 2010.

[Motik and Rosati, 2010] Boris Motik and Riccardo Rosati. Reconciling Description Logics and Rules. *Journal of the ACM*, 57(5):1–62, June 2010.

[Pührer *et al.*, 2010] Jörg Pührer, Stijn Heymans, and Thomas Eiter. Dealing with inconsistency when combining ontologies and rules using DL-programs. In *Proceedings of 7th Extended Semantic Web Conference, part I*, pages 183–197, Heraklion, Crete, Greece, May-June 2010. Springer.

[Sakama and Inoue, 2003] Chiaki Sakama and Katsumi Inoue. An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming*, 3(6):671–713, May 2003.

[Shen, 2011] Yi-Dong Shen. Well-supported semantics for description logic programs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1081–1086, Barcelona, Catalonia, Spain, July 2011. IJCAI/AAAI.

[Wang *et al.*, 2010] Yisong Wang, Jia-Huai You, Li-Yan Yuan, and Yi-Dong Shen. Loop formulas for description logic programs. *Theory and Practice of Logic Programming*, 10(4-6):531–545, July 2010.