ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-381

Schema-Based Debugging of Federated Data Sources

Andreas Nolle[1] and **Christian Meilicke**[2] and **Melisachew Wudage Chekol**[2] and **German Nemirovski**[1] and **Heiner Stuckenschmidt**[2]

Abstract. Information explosion leads to continuous growth of data distributed over different data sources. However, the increasing number of data sources increases the risk of inconsistency. In such a federative setting, description logics can be applied to define a central schema that serves as a conceptual view comprising and extending the semantics of each data source. Consequently, each data source is treated as a single knowledge base that is integrated in a federated knowledge base. Following this idea, we propose an approach for automated debugging of federated knowledge bases that targets the identification and repair of inconsistency. We report on experiments with a large distributed dataset from the domain of library science.

1 Introduction

The Linked Open Data (LOD) cloud grows continuously. However, the more data is available, the higher is the probability of inconsistency. Besides local clashes within data sources, two (or more) data sources can be contradictory. In the context of library science, for instance, one data source may catalogue a publication correctly as a paper, whereas in another source it is mistakenly defined as proceedings. One approach to tackle this problem is the use of federated data or information integration, where a central schema serves as a conceptual view that comprises and extends the semantics of each integrated data source. As a consequence, the central schema and its mappings to the different schemas which are used in the integrated data sources represent the interface to the distributed data. By using these mappings, original queries (mainly referring to the central schema) can be transformed into queries referring to the related schema of each data source. Thus, clients do not have to be aware of the local schema in each integrated data source [24].

Let us illustrate this by the following example. We will use this example throughout the remainder of the paper.

Example 1 *Let $\mathcal{T}$ be a central schema and $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ denote three distributed data sources. $\mathcal{T}$ contains the following axioms.*

$$Book \sqcup Paper \sqsubseteq Publication \qquad Paper \sqsubseteq \neg Book$$
$$Proceedings \sqsubseteq Book \qquad Book \sqcup Paper \sqsubseteq \neg SlideSet$$
$$\exists isPartOf \sqsubseteq Paper \qquad \exists isPartOf^{-} \sqsubseteq Proceedings$$

The three data sources contain the following assertions:

$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
$Paper(\mathbf{I1})$	$Paper(\mathbf{I1})$	$SlideSet(\mathbf{I1})$
$isPartOf(\mathbf{I1}, \mathbf{AI15})$	$Proceedings(\mathbf{I1})$	$SlideSet(\mathbf{I2})$
$Paper(\mathbf{I2})$	$isPartOf(\mathbf{AI15}, \mathbf{I1})$	

[1] Albstadt-Sigmaringen University, Germany, email: {nolle, nemirovskij}@hs-albsig.de
[2] Research Group Data and Web Science, University of Mannheim, Germany, email: {christian, mel, heiner}@informatik.uni-mannheim.de

The assertion that $\mathbf{I1}$ is a $Paper$ (in $\mathcal{A}_1$) and the assertion that $\mathbf{I1}$ is a $SlideSet$ (in $\mathcal{A}_3$) are obviously in contradiction due to the axiom $Paper \sqsubseteq \neg SlideSet$ in $\mathcal{T}$. In addition, as the assertion $Paper(\mathbf{I1})$ can also be found in $\mathcal{A}_2$, it is also contradictory to $\mathcal{A}_3$. Furthermore, we can entail this assertion in $\mathcal{A}_1$ from $isPartOf(\mathbf{I1}, \mathbf{AI15})$ and the axiom $\exists isPartOf \sqsubseteq Paper$ in $\mathcal{T}$.

Note that our example can easily be extended to the case where the integrated data sources use different terminologies that are linked by equivalence or subsumption axioms to an intermediary schema. Without loss of generality, we will in the remainder of this paper assume that there is only one central schema $\mathcal{T}$ which might be the union of some data source specific schemas and an intermediary one containing mappings between the data source specific vocabularies. Furthermore, in our work, we will not address integration problems related to the incoherency of $\mathcal{T}$, i.e., we assume that $\mathcal{T}$ is coherent. Note that there are other works that deal with debugging issues on the terminological level, e.g., [10].

The main contribution of our approach is to exploit explicit but also implicit redundancies caused by federating different sources to verify or disprove assertions that are involved in logical conflicts and to propose a resolution of these conflicts. In a setting with two or more data sources, where each data source contains several thousand assertions, it is challenging to propose a solution that takes the dependencies between the involved conflicts in an appropriate way into account. Based on techniques like query expansion (backward-chaining) and by identifying inconsistencies via clash queries we first collect all logical conflicts then we apply a two-phase debugging algorithm to resolve the previously collected conflicts.

In particular, we apply a majority voting scheme. Based on this approach we are able to resolve a subset of all conflicts in the first phase of our debugging algorithm. The second phase uses the outcome of the first phase to deduce a data source specific measure of trust for certain types of assertions. Repairs of additional conflicts can then be generated based on the statistical evidences gathered in the first phase. We argue why our algorithm generates reasonable repair plans and evaluate our approach against a large distributed LOD dataset from the domain of library science.

The rest of the paper is organized as follows. In Section 2 we introduce *DL-Lite*$_\mathcal{A}$ as well as some fundamental terms and definitions related to inconsistency detection in federated *DL-Lite*$_\mathcal{A}$ knowledge bases and conjunctive queries. In Section 3 we recall and extend our previous approach of inconsistency detection in federated *DL-Lite*$_\mathcal{A}$ knowledge bases. Subsequently, we propose our algorithm for the generation of repairs in Section 4 comprising the two phases of resolvable and learned repairs. In Section 5 we discuss some evaluation results of our experiments with a large distributed LOD dataset. Before concluding in Section 7, we compare approaches related to our work in Section 6.

2 Preliminaries

We briefly introduce our definition of federated *DL-Lite*$_\mathcal{A}$ knowledge bases (KBs), discuss basic notions related to inconsistency in DL KBs, and describe conjunctive queries over *DL-Lite*$_\mathcal{A}$ KBs.

2.1 Federated *DL-Lite*$_\mathcal{A}$ Knowledge Bases

DL-Lite is a family of lightweight description logics proposed by Calvanese et al. [4] with the aim to find a trade-off between expressiveness and reasoning complexity. This resulted in a family of languages comprising various *DL-Lite* logics where reasoning, such as traditional DL reasoning services like checking KB satisfiability, can be done in PTIME in the of size of the TBox and query answering in AC^0 in the size of the ABox. Furthermore, it has been shown that members of the *DL-Lite* family are one of the maximal logics that allow first-order logic (FOL)-rewritability of conjunctive query answering and therewith a processing of query answering through standard database technology. For this study, we consider the sub-family *DL-Lite*$_\mathcal{A}$, which has been especially designed for dealing efficiently with huge amounts of extensional information.

In *DL-Lite*$_\mathcal{A}$ concept, role, value-domain, and attribute expressions are formed according to the following syntax:

$$\begin{array}{ll} B ::= \bot_C \mid A \mid \exists Q \mid \delta(U) & E ::= \rho(U) \\ C ::= \top_C \mid B \mid \neg B \mid \exists Q.C & F ::= \top_D \mid T_1 \mid \ldots \mid T_n \\ Q ::= P \mid P^- & V ::= U \mid \neg U \\ R ::= Q \mid \neg Q & \end{array}$$

where $\top_C$ denotes the *top* or *universal concept*, $\bot_C$ the *bottom* or *empty concept*, *A* an *atomic concept*, *B* a *basic concept* and *C* a *general concept*. Similar to that, we have *atomic roles* denoted by *P*, *basic roles* by *Q* and *general roles* by *R*. *Atomic attributes* are represented by *U* and *general attributes* by *V* whereas *E* denotes a *basic value-domain* and *F* a *value-domain expression*. Furthermore, $\exists Q$ (*unqualified existential restrictions*) represent objects that are related by role *Q* to some objects, $\exists Q.C$ (*qualified existential restrictions*) denote objects that are related by *Q* to objects denoted by concept *C*, $\neg$ denotes the negation of concepts, roles or attributes and P^- is used to represent the inverse of role *P*. Concerning an attribute *U* its *domain* is denoted by $\delta(U)$ and its *range* (set of values) by $\rho(U)$. *Value domains* are represented by $T_1 \mid \ldots \mid T_n$, where each T_i denotes a pairwise disjoint datatype of values and $\top_D$ the *universal value-domain* [4, 18]. In *DL-Lite*$_\mathcal{A}$ a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a TBox $\mathcal{T}$ also known as schema, and an ABox $\mathcal{A}$, the extensional knowledge part which represents a data source.

The TBox $\mathcal{T}$ contains a set of axioms of the form

$$B \sqsubseteq C \quad Q \sqsubseteq R \quad E \sqsubseteq F \quad U \sqsubseteq V \quad (\mathsf{funct}\ Q) \quad (\mathsf{funct}\ U)$$

and the ABox $\mathcal{A}$ is a finite set of assertions of the form

$$A(a) \quad P(a,b) \quad U(a,v).$$

TBox assertions of the form $B \sqsubseteq C$ denotes *concept inclusions*, $Q \sqsubseteq R$ *role inclusion*, $E \sqsubseteq F$ *value-domain inclusion* and $U \sqsubseteq V$ *attribute inclusion*. *Functionality assertions* on roles and attributes in $\mathcal{T}$ are denoted by (funct *Q*) and (funct *U*), respectively. TBox assertions of the form $B_1 \sqsubseteq B_2$ and $Q_1 \sqsubseteq Q_2$ are called *positive inclusions* (*PI*) whereas $B_1 \sqsubseteq \neg B_2$ and $Q_1 \sqsubseteq \neg Q_2$ *negative inclusions* (*NI*). For ABox assertions *a* and *b* represent object constants and *v* represents a value constant.

The semantics of *DL-Lite*$_\mathcal{A}$ is given in terms of an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ (the domain) is a disjoint union of the two non-empty sets $\Delta^\mathcal{I}_\mathcal{O}$, the domain of objects, and $\Delta^\mathcal{I}_\mathcal{V}$, the domain of values; and $\cdot^\mathcal{I}$ (the interpretation function) that maps each element in the signature Σ (also known as alphabet or vocabulary) to a subset of $\Delta^\mathcal{I}_\mathcal{O}$ and each value domain to a subset of $\Delta^\mathcal{I}_\mathcal{V}$. *DL-Lite*$_\mathcal{A}$ adopts the unique name assumption (UNA), meaning that for every interpretation $\mathcal{I}$ and constant pair $c_1 \neq c_2$, we have $c_1^\mathcal{I} \neq c_2^\mathcal{I}$. This means that different constant names (encoded as IRIs) are interpreted differently and refer to different individuals. In terms of further semantics we refer to the more precise definitions given in [4, 18].

In the context of federated settings, where each integrated data source uses different terminologies that are linked by an intermediary (central) schema, we can define a federated *DL-Lite*$_\mathcal{A}$ KB as well as federated ABox assertions as follows:

Definition 1 *A federated DL-Lite*$_\mathcal{A}$ *knowledge base is a DL-Lite*$_\mathcal{A}$ *knowledge base* $\mathcal{K}$ *with* $\mathcal{K} = \langle \mathcal{T}_c \cup \bigcup_{i\in\mathbb{F}} \mathcal{T}_i, \bigcup_{i\in\mathbb{F}} \mathcal{A}_i \rangle$ *where* $\mathcal{T}_c$ *is a central TBox, each* $\mathcal{T}_i$ *is a TBox and* $\mathcal{A}_i$ *is an ABox in data source* i *and* $\mathbb{F}$ *is a set of indices that refers to the federated data sources. A federated ABox assertion is a pair* $\langle \alpha, i \rangle$ *where* α *denotes an ABox assertion stated in* $\mathcal{A}_i$*. For compact presentation we will write only* $\mathcal{T}$ *instead of* $\mathcal{T}_c \cup \bigcup_{i\in\mathbb{F}} \mathcal{T}_i$ *and* $\mathcal{A}$ *instead of* $\bigcup_{i\in\mathbb{F}} \mathcal{A}_i$ *for the rest of this paper.*

2.2 Inconsistency in Description Logics

In description logics, an interpretation $\mathcal{I}$ that satisfies all KB assertions in $\mathcal{T} \cup \mathcal{A}$ is called a *model*. The set of all models of $\mathcal{K}$ is denoted by *Mod*($\mathcal{K}$) and if *Mod*($\mathcal{K}$) $\neq \emptyset$, we call $\mathcal{K}$ *satisfiable* or *consistent* [2, 7]. Otherwise $\mathcal{K}$ is called *inconsistent*. $\mathcal{K} \models \phi$ denotes that $\mathcal{K}$ logically entails or satisfies a closed first-order logic sentence (formula) ϕ, if $\phi^\mathcal{I}$ is true for every $\mathcal{I} \in$ *Mod*($\mathcal{K}$). If a set of closed sentences denoted by *F* is entailed by $\mathcal{K}$, we can also write $\mathcal{K} \models F$ [21]. According to Kalyanpur et al. [11] an *explanation* (or justification) for $\mathcal{K} \models \phi$ is a subset $\mathcal{K}'$ of $\mathcal{K}$ such that $\mathcal{K}' \models \phi$ while $\mathcal{K}'' \not\models \phi$ for all $\mathcal{K}'' \subset \mathcal{K}'$. An explanation can be understood as a minimal reason that explains why ϕ follows from $\mathcal{K}$. Analogously, given an inconsistent knowledge base $\mathcal{K}$, we are interested in explanations for the inconsistency, i.e., minimal subsets $\mathcal{K}'$ of $\mathcal{K}$ such that *Mod*($\mathcal{K}'$) $= \emptyset$. More precisely, a minimal inconsistent subset (*MIS*) denoted by $\mathcal{K}'$ is a subset of $\mathcal{K}$ such that $\mathcal{K}'$ is inconsistent while $\mathcal{K}''$ is consistent for all $\mathcal{K}'' \subset \mathcal{K}'$. From our running example, we can see that $\langle \mathcal{T}, \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \rangle$ is an inconsistent KB.

Example 2 *One of the explanations for the inconsistency mentioned in Example 1 is the set*

$$\{isPartOf(\mathbf{I1}, \mathbf{AI15}), SlideSet(\mathbf{I1}), \\ Paper \sqsubseteq \neg SlideSet, \exists isPartOf \sqsubseteq Paper\}.$$

A subset $\mathcal{R} \subseteq \mathcal{K}$ is called a *repair* (or repair plan) of $\mathcal{K}$, if $\mathcal{K}$ is inconsistent and if $\mathcal{K} \setminus \mathcal{R}$ is consistent. As shown in [20], a hitting set over all MISs is a repair. Note that there is always a trivial repair $\mathcal{R} = \mathcal{K}$. However, we are especially interested in those repairs that remove a minimal number of assertions, i.e., $\mathcal{R}$ is a minimal repair if $\mathcal{R}$ is a repair and each proper subset of $\mathcal{R}$ is not a repair.

2.3 Conjunctive Queries

A *conjunctive query* (CQ) *q* over a KB $\mathcal{K}$ is a Datalog expression of the form $q(\boldsymbol{x}) \leftarrow conj(\boldsymbol{x}, \boldsymbol{y})$. *conj*($\boldsymbol{x}$,$\boldsymbol{y}$) denotes the *body* of *q* and is a conjunction of *atoms* of the form $A(x)$, $P(x,y)$, $x = y$, or $x \neq y$ in which *x* and *y* are either constants in $\mathcal{K}$ or variables in $\boldsymbol{x}$ or $\boldsymbol{y}$, and *A* is a concept name or value-domain in $\mathcal{K}$ and *P* is a role or attribute

name in $\mathcal{K}$. In addition, $\boldsymbol{x}$ are *distinguished variables* that are part of the *head* $q(\boldsymbol{x})$ of a query q whereas $\boldsymbol{y}$ are *non-distinguished variables* and do not occur in the head. If a variable does not correspond to the set of distinguished variables and does not occur in at least two query atoms, the variable is called *unbound* and is denoted by _. *Unions of conjunctive queries* (UCQ) are denoted by the expressions $q(\boldsymbol{x}) \leftarrow conj_1(\boldsymbol{x}, \boldsymbol{y}_1), \ldots, q(\boldsymbol{x}) \leftarrow conj_n(\boldsymbol{x}, \boldsymbol{y}_n)$, where each $conj_i(\boldsymbol{x}, \boldsymbol{y}_i)$ is a conjunctive query.

Example 3 *The following query, over the KB in Example 1, selects all papers that have been published in proceedings:*

$$q(x) \leftarrow Paper(x), isPartOf(x, \_).$$

3 Inconsistency Detection

This section recalls our previous approach of efficiently detecting inconsistency in federated KBs as first presented in [17]. We extend this approach and focus on the generation of federated explanations.

3.1 Inconsistency Detection in *DL-Lite*$_\mathcal{A}$

To determine if a KB is consistent or not, we have to search for ABox assertions, that are in conflict with the TBox or that are contradicting each other given the TBox. Lembo et al. [14] identified a complete set of six different patterns that cause clashes in *DL-Lite*$_\mathcal{A}$ KBs:

(a) an instantiation of an unsatisfiable (incoherent) concept, role or attribute such that $\mathcal{T} \models A \sqsubseteq \neg A$ and $A(a) \in \mathcal{A}$ (respectively $\mathcal{T} \models P \sqsubseteq \neg P$ and $P(a, b) \in \mathcal{A}$ for roles, and $\mathcal{T} \models U \sqsubseteq \neg U$ and $U(a, v) \in \mathcal{A}$ for attributes)

(b) ABox assertions contradicting axioms that restrict the interrelation of individuals such that $\mathcal{T} \models P \sqsubseteq \neg P^-$ or $\mathcal{T} \models \exists P \sqsubseteq \neg \exists P^-$ and $P(a, a) \in \mathcal{A}$

(c) incorrect datatypes such that $\mathcal{T} \models \rho(U) \sqsubseteq T$, $U(a, v) \in \mathcal{A}$ and $v^\mathcal{I} \notin T^\mathcal{I}$

(d) ABox assertions contradicting negative inclusions such that, e.g., $\mathcal{T} \models A \sqsubseteq \neg \exists P$ and $\{A(a), P(a, b)\} \subseteq \mathcal{A}$

(e) ABox assertions contradicting role functionality such that (funct P) $\in \mathcal{T}$ and $\{P(a, b_1), P(a, b_2)\} \subseteq \mathcal{A}$, where $b_1 \neq b_2$ (respectively (funct P^-) $\in \mathcal{T}$ and $\{P(a_1, b), P(a_2, b)\} \subseteq \mathcal{A}$, where $a_1 \neq a_2$, for the functionality of a inverse role)

(f) ABox assertions contradicting attribute functionality such that (funct U) $\in \mathcal{T}$ and $\{U(a, v_1), U(a, v_2)\} \subseteq \mathcal{A}$, where $v_1 \neq v_2$

The distribution of huge amounts of data over several sources makes state of the art methods for inconsistency detection, such as tableau-based reasoning algorithms, hardly applicable (see [17]), since they mostly require to have all the data in one place. We propose an alternative approach, comprising the formulation and evaluation of *federated clash queries*.

3.2 Clash Query Generation

Based on the clash definitions given above and referring to the work of Calvanese et al. [4] we define a mapping function φ to generate queries for inconsistency detection out of relevant axioms from $\mathcal{T}$. The function φ maps concepts, roles and attributes into query atoms as follows:

$$\begin{array}{lll} A \mapsto A(x) & \delta(U) \mapsto U(x, \_) & P \mapsto P(x, y) \\ \exists P \mapsto P(x, \_) & \rho(U) \mapsto U(\_, y) & P^- \mapsto P(y, x) \\ \exists P^- \mapsto P(\_, x) & T \mapsto T & \end{array}$$

Based on φ, the clash types (a)–(f) can be mapped into queries (i)–(vi) as shown below:

(i) $\varphi(A \sqsubseteq \neg A) = q(x) \leftarrow \varphi(A)$,
$\varphi(X \sqsubseteq \neg X) = q(x, y) \leftarrow \varphi(X)$, where $X \in \{P, U\}$,

(ii) $\varphi(P \sqsubseteq \neg P^-) = q(x, y) \leftarrow \varphi(P), \varphi(P^-)$ and
$\varphi(\exists P \sqsubseteq \neg \exists P^-) = q(x) \leftarrow \varphi(\exists P), \varphi(\exists P^-)$,

(iii) $\varphi(\rho(U) \sqsubseteq T) = q(x, y) \leftarrow U(x, y), datatype(y) \neq T$,

(iv) $\varphi(C \sqsubseteq \neg D) = q(x) \leftarrow \varphi(C), \varphi(D)$, where $C, D \in \{A, \exists P, \exists P^-, \delta(U)\}$,
$\varphi(R \sqsubseteq \neg S) = q(x, y) \leftarrow \varphi(R), \varphi(S)$ and
$\varphi(V_1 \sqsubseteq \neg V_2) = q(x, y) \leftarrow \varphi(V_1), \varphi(V_2)$,

(v) $\varphi(\text{funct } P) = q(x, y, z) \leftarrow P(x, y), P(x, z), y \neq z$ and
$\varphi(\text{funct } P^-) = q(x, y, z) \leftarrow P(y, x), P(z, x), y \neq z$, and

(vi) $\varphi(\text{funct } U) = q(x, y, z) \leftarrow U(x, y), U(x, z), y \neq z$,

where A, P and U denote an atomic concept, an atomic role, and an atomic attribute; $x, y, z, \_$ are variables; $\exists P, \exists P^-$ and $\delta(U)$ are concepts; R and S are roles; V_1 and V_2 are attributes; $datatype(y)$ is an external function which computes the datatype of a given data value y; and T denotes a datatype of values, where each different datatypes are pairwise disjoint. Except for the clash queries in (i), the queries in (ii)–(vi) contain two atoms and an inequality constraint in (v) and (vi) used as filters applied to the query answers. We refer to the queries in (ii)–(vi) as *two-atom* queries.

Example 4 *For the axiom $Paper \sqsubseteq \neg Book$ in Example 1, the mapping function φ generates the clash query: $\varphi(Paper \sqsubseteq \neg Book) = q(x) \leftarrow Paper(x), Book(x)$.*

According to the definition above, the mapping function φ generates UCQs that may contain inequalities (because of clash queries in (v) and (vi)), that in general makes query answering intractable. However, since those queries are of fixed length (two atoms and an inequality expression), the complexity of checking KB satisfiability by a reduction into query answering is in AC^0 in the size of the ABox and NLOGSPACE in the size of the KB as shown by Artale et al [1].

3.3 Clash Query Expansion

To ensure that all implicit knowledge is taken into consideration when computing the answers, the original query is expanded. The resulting set of expanded queries (UCQs) will contain atoms addressing all possible concepts, roles and attributes that implicitly provide individuals of the originally requested type. For *DL-Lite*$_\mathcal{A}$ KBs query expansion (backward-chaining) of a general UCQ can be efficiently done in PTIME in the size of the TBox [4].

Definition 2 *Given a TBox $\mathcal{T}$ and a query $q(\boldsymbol{x})$ in the signature of $\mathcal{T}$. An expansion of $q(\boldsymbol{x})$ is a UCQ denoted by $q_{exp}(\boldsymbol{x}) = \bigcup_i q_i(\boldsymbol{x})$, that is a rewriting of $q(\boldsymbol{x})$ w.r.t. $\mathcal{T}$, such that $\langle \mathcal{T}, \mathcal{A} \rangle \models q(\boldsymbol{a})$ iff $\mathcal{A} \models q_{exp}(\boldsymbol{a})$, for any ABox $\mathcal{A}$ and any tuple $\boldsymbol{a}$ of individuals in $\mathcal{A}$.*

Example 5 *The expansion of the clash query in Example 4 is the following UCQ:*

$$\begin{aligned} q_{exp}(x) &\leftarrow Paper(x), Book(x), \\ q_{exp}(x) &\leftarrow Paper(x), Proceedings(x), \\ q_{exp}(x) &\leftarrow Paper(x), isPartOf(\_, x), \\ q_{exp}(x) &\leftarrow isPartOf(x, \_), Proceedings(x), \\ q_{exp}(x) &\leftarrow isPartOf(x, \_), Book(x), \\ q_{exp}(x) &\leftarrow isPartOf(x, \_), isPartOf(\_, x) \end{aligned}$$

Subsumption axioms in *DL-Lite*$_\mathcal{A}$ comprise only one element on the left and one element on the right hand side, or can be normalized to that form. Consequently, an expansion of a clash query is a UCQ

where each conjunct has one or at most two query atoms and an inequality constraint.

Since q_{exp} may comprise query atoms containing unbound variables, we replace those variables by new distinguished variables so as to make precise distinctions between different instantiations of them.

3.4 Generating Federated Explanations

Computing all inconsistency explanations, introduced earlier as MIS, requires to generate and expand all clash queries. The resulting UCQs comprise not only the semantics of the central TBox but also of each integrated data source. Thus, each atom of those queries may address several data sources. Local MISs are detected by evaluating the expanded queries directly against a data source. Federated MISs are detected by distributing the atoms of the expanded queries as atomic queries to the data sources. Consequently, we apply a simple federation algorithm where each query atom is evaluated at all data sources.

Example 6 *According to that, the following federated queries will be generated for the expanded query q_{exp}(x,y) ← Paper(x), isPartOf(y,x) (third conjunct of Example 5):*

$$q_{exp}(x,y) \leftarrow Paper(x), isPartOf(y,x)$$

$\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ $\qquad$ $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$

$q'(x) \leftarrow Paper(x)$ $\qquad$ $q''(x,y) \leftarrow isPartOf(y,x)$

The results of these queries are tuples of instances. However, we are not interested in tuples of instances, but in the inconsistency explanations (MIS) that can be derived from these results. Because of that, the results of query pairs have to be joined and converted to federated ABox assertions (see Example 7). Moreover, since we assume that all of the terminological axioms are correct, we are only targeting the subset of a MIS that contain only ABox assertions. In the following we refer to such a subset of a MIS as a MISA (minimal inconsistency preserving sub-ABox).

Example 7 *The set of MISAs resulting from the evaluation of the query in Example 6 is the set*

$$\{\{\langle Paper(\mathbf{I1}),1\rangle, \langle isPartOf(\mathbf{AI15},\mathbf{I1}),2\rangle\},$$
$$\{\langle Paper(\mathbf{I1}),2\rangle, \langle isPartOf(\mathbf{AI15},\mathbf{I1}),2\rangle\}\}.$$

None of these operations has an impact on the complexity which remains in AC^0 in the size of the ABox and in NLOGSPACE in the size of the whole KB, given a fixed set of data sources.

4 Repair Plan Generation

The generation of a repair plan is divided into two phases. In the first phase we propose a partial repair plan following a simple majority voting scheme (Section 4.1), while in the second phase (Section 4.2) we try to repair the remaining conflicts following an approach guided by the statistics gathered in the first phase.

4.1 Phase 1: Majority Voting

To resolve the identified contradictions we follow the assumption that the more data sources are integrated, the higher is the probability that correct assertions occur redundantly. Conversely, the probability that an assertion is incorrect correlates with the number of contradictions in which the assertion is involved.

Based on this assumption, we propose a greedy approach, given in Algorithm 1, for generating repairs. The algorithm starts with the trivial repair of a singleton MISA (resulting from clash type (b) or (c)) by removing the only element in each singleton MISA. This repair can also have an influence on the remaining steps, because the element of a singleton MISA might also appear in a MISA with two elements. The remaining part of the algorithm deals with a non trivial repair of MISA with two elements. In the main loop the algorithm first counts for each assertion in how many different MISAs it occurs. We refer to the resulting number as a *cardinality of an assertion*. We also compute the *cardinality of a MISA* which is defined as the sum of the cardinalities of its two elements. We call MISAs that have elements with different cardinalities *resolvable MISAs*. With the help of a majority voting heuristic, we can make a decision in favour of one of the two elements of a resolvable MISA. We select all resolvable MISAs with minimum cardinality and remove from these MISAs the element with higher cardinality, which is the element that is involved in more conflicts. Note that we resolve MISAs with minimum cardinality first, to reduce the impact (of wrong decisions) on subsequent decisions. After each removal operation we update the remaining set of MISAs and repeat this procedure as long as resolvable MISAs can be found. The algorithm terminates when no resolvable MISAs are left to be repaired.

Algorithm 1: GenerateResolvableRepairs($\mathcal{C}$)

```
Output: (partial) repair R resolved by majority voting
begin
    R ← ∅
    C_unary ← GetSingletonMISA(C)
    foreach c ∈ C_unary do
        R ← R ∪ GetAssertion(c)
        C_resolved ← GetResolvedMISAs(GetAssertion(c), C)
        C ← C\C_resolved
    while true do
        C_card ← DetermineCardinalities(C)
        C_x ← GetResolvableMISAsWithMinCard(C_card)
        if C_x = ∅ then
            break
        foreach c ∈ C_x do
            α ← GetAssertionWithMaxCard(c, C_card)
            R ← R ∪ α
            C_resolved ← GetResolvedExplanations(α, C)
            C ← C\C_resolved
    return R
end
```

The algorithm is based on a heuristics that selects edges (MISAs) with minimal weight and removes from such edges the vertex (assertion) that is involved in more MISAs. Our algorithm runs in polynomial time with respect to the number of vertices. It does not guarantee, even for the resolvable cases, to construct a minimal vertex cover. The construction of a minimal vertex cover is known to be one of Karps NP-complete problems [12].

Example 8 *In our running example we have to deal with the federated assertions α_1 to α_8 listed as follows.*

$\alpha_1 = \langle Paper(\mathbf{I1}),1\rangle$ $\qquad$ $\alpha_2 = \langle isPartOf(\mathbf{I1},\mathbf{AI15}),1\rangle$

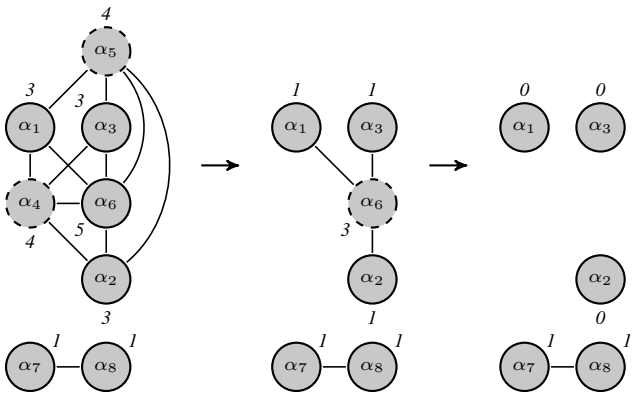
$\alpha_3 = \langle Paper(\mathbf{I1}),2\rangle$ $\qquad$ $\alpha_4 = \langle Proceedings(\mathbf{I1}),2\rangle$

$\alpha_5 = \langle isPartOf(\mathbf{AI15},\mathbf{I1}),2\rangle$ $\qquad$ $\alpha_6 = \langle SlideSet(\mathbf{I1}),3\rangle$

$\alpha_7 = \langle Paper(\mathbf{I2}),1\rangle$ $\qquad$ $\alpha_8 = \langle SlideSet(\mathbf{I2}),3\rangle$

The set of MISAs $\mathcal{C}$ for this example is $\{\{\alpha_1,\alpha_4\}, \{\alpha_1,\alpha_5\}, \{\alpha_1,\alpha_6\}, \{\alpha_2,\alpha_4\}, \{\alpha_2,\alpha_5\}, \{\alpha_2,\alpha_6\}, \{\alpha_3,\alpha_4\}, \{\alpha_3,\alpha_5\}, \{\alpha_3,\alpha_6\}, \{\alpha_4,\alpha_6\}, \{\alpha_5,\alpha_6\}, \{\alpha_7,\alpha_8\}\}$. *The assertion α_1 has a*

cardinality of three because it appears in three MISAs; assertion α_4 has a cardinality of four. Thus, MISA $\{\alpha_1, \alpha_4\}$ has a cardinality of seven. In the following we represent each assertion as a vertex in a graph where each MISA is represented by an edge. This graph is shown in the following to illustrate the iterations of Algorithm 1. Note that we annotated the graph with assertion cardinalities, but omitted MISA cardinalities due to the lack of space.



As shown in the figure above, the algorithm needs three iterations to construct a repair, i.e., to construct a vertex cover for the corresponding conflict graph. Consequently, the resolvable repairs of our running example comprise α_4, α_5 and α_6. Assertion α_7 and α_8 yield an inconsistency, however, they are not conflicting with the other assertions. This inconsistency cannot be resolved by Algorithm 1, because α_7 and α_8 have the same cardinality which is one.

As illustrated in Example 8, the algorithm cannot resolve all logical conflicts. This will be the case, especially when the set of MISAs contains some MISAs that are unresolvable (by comprising elements having the same cardinalities) and are also not resolved during the process of executing Algorithm 1 due to an overlap with a resolvable MISA. Particularly, contradictory assertions of different values for a functional role or attribute result in MISAs that are predestinated to be unresolvable. How to deal with the remaining clashes is explained in the next section.

4.2 Phase 2: Learned Repairs

In the second phase we use the statistical evidence, that is implicitly available in the repair computed so far, to extend this repair. Suppose that a large fraction of the assertions of type $C(x)$ have been removed from a data source A_i, while most of the assertions $D(x)$ in A_j have not been removed. Now suppose that we have a MISA $\{\langle C(a), i\rangle, \langle D(b), j\rangle\}$. If we trust in the correctness of the repair that we conducted so far, we are justified in removing $\langle C(a), i\rangle$, because we have a higher trust in $\langle D(b), j\rangle$. Let us introduce the notion of trust formally.

Definition 3 *Given a federated knowledge base $\mathcal{K} = \langle \mathcal{T}, \bigcup_{i\in\mathbb{F}} \mathcal{A}_i\rangle$, and a repair $\mathcal{R}$ computed by Algorithm 1. Let σ be either a concept, a property or an attribute in the signature Σ of $\mathcal{T}$, and let $\Psi \subseteq \bigcup_{i\in\mathbb{F}} \mathcal{A}_i$ be a set of federated assertions, then $sas(\sigma, \Psi, i)$ is defined as the subset of assertions in Ψ that use σ and originate from $\mathcal{A}_i$. The trust in σ with respect to i is defined as*

$$trust(\sigma, i) = 1 - \frac{|sas(\sigma, \mathcal{R}, i)|}{\left|sas(\sigma, \bigcup_{i\in\mathbb{F}} \mathcal{A}_i, i)\right|}.$$

Based on this definition we define the trust of a federated assertion $\langle \alpha, i\rangle$ that uses σ as $trust(\langle \alpha, i\rangle) = trust(\sigma, i)$.

Example 9 *From the repair of Example 8 it follows that we have $trust(Paper, 1) = 1 - \frac{0}{2} = 1$ and $trust(SlideSet, 3) = 1 - \frac{1}{2} = 0.5$. Thus, we remove α_8 as a* learned repair *due to the fact that α_7 has a higher trust.*

This example shows how to apply the notion of trust to a single MISA, however, the set of all remaining MISAs might still contain overlapping MISAs. Therefore, we have to implement it as part of a more general algorithm. In [19] the authors proposed a linear algorithm for debugging terminological alignments. The proposed algorithm can be applied to any debugging scenario where a complete set of conflict sets, in our case the set of MISAs, is given. The algorithm, which we sketch in the following, requires a complete ordering of assertions that are involved in the remaining clashes. We derive this ordering from the trust values.

The input to Algorithm 2 are the unresolved MISAs $\mathcal{C}$, the previously computed repair $\mathcal{R}$, and a trust-ordered list $\mathcal{A}_{trust}$ of all assertions that occur in $\mathcal{C}$. The algorithm iterates over $\mathcal{A}_{trust}$ in descending order, thus, starting with an assertion α for which there is no assertion with higher trust. In each iteration the algorithm determines all those MISAs that contain α. For each such MISA $\{\alpha, \beta\}$ the assertion β is added to the repair $\mathcal{R}'$ if the trust of β is lower than the trust of α. Thus, we finally remove β for the reason that we presented at the beginning of this section.

Algorithm 2: GenerateLearnedRepairs($\mathcal{C}$, $\mathcal{R}$, $\mathcal{A}_{trust}$)

```
Output: all learned repairs R'
begin
    R' ← ∅
    foreach α ∈ A_trust do
        C' ← {{x, y} ∈ C | α ∈ {x, y}}
        foreach {x, y} ∈ C' do
            β ← x if x ≠ α otherwise y
            if trust(α) > trust(β) then
                R' ← R ∪ β
                C_resolved ← {{x', y'} ∈ C | β ∈ {x', y'}}
                C ← C \ C_resolved
    return R'
end
```

Algorithm 2 runs in the worst case in quadratic time with respect to the number of vertices $\mathcal{C}$ (unresolved MISAs). Thus, the complexity of the whole federated debugging approach, starting from the generation of the clash queries up to the the generation of resolvable and learned repairs, runs in polynomial time.

Note that the algorithm will resolve all clashes if there is no MISA comprising two assertions with the same trust value. However, this will not always be the case. The repair of still remaining MISAs is not addressed in this paper. A possible extension of our approach could be the calculation of a general trust value for each data source over all of its assertions. Alternatively, a user could decide upon the problematic cases.

5 Experimental Evaluation

In order to evaluate the effectiveness of our approach we have set up a large distributed LOD dataset from the domain of library science. Specifically, we selected four LOD data sources, referred to as $\mathcal{A}_1$ to $\mathcal{A}_4$ in the following, and loaded their dumps into separate Virtuoso 7.2.2 instances (Open-Source Edition). In particular, these data

sources are FacetedDBLP[3] ($\mathcal{A}_1$), BibSonomy[4] ($\mathcal{A}_2$), RKB Explorer ePrints Open Archives[5] ($\mathcal{A}_3$), and RKB Explorer DBLP[6] ($\mathcal{A}_4$).

Note that our implementation relies on the usage of standard SPARQL interfaces and does not put any additional requirements on the data sources. Since the OWL 2 QL profile is based on *DL-Lite*, we have used it as specification language of our central TBox that includes the TBoxes of each data source. Note that we have applied some small modifications of the data source specific TBoxes to ensure that the federated TBox is coherent. Since the federated TBox lacks some negative inclusions and functionality assertions, we have added respective axioms to the central TBox.

In contrast to *DL-Lite*$_\mathcal{A}$ (see Section 2.1), the standard ontology language OWL, i.e., the OWL 2 QL profile, does not make the UNA, however, OWL provides the explicit object property `owl:sameAs` to express that two IRIs denote the same individual. Due to the fact that LOD sources following this strategy to link same individuals, we took the `owl:sameAs` assertions into account and modified our dataset such that all individuals representing the same entity also have the same IRI. Note that according to the work of Calvanese et al. [6] it is, under some restrictions, even possible to take into account `owl:sameAs` statements for query answering and retain the FOL-rewritability. But on grounds of simplicity of our experimental evaluation we embark on the strategy of resolving linked individuals by modifying the datasets. In addition to that, to gain a higher overlapping of the data sources we detected duplicates, especially by the unique attributes denoting the ISBN or the ISSN of a publication. The collection of the central TBox as well as the referenced TBoxes is available online[7]. For legal reasons we are currently not able to publish the final dataset of each integrated data source. Please contact us if you are interested in these datasets.

Based on the federated TBox our algorithm generates 422 clash queries, where 8 of which result from functionality assertions and 414 result from negative inclusions. Since some of those clash queries, i.e., the queries resulting from negative inclusions, can be implicitly derived by another clash query, the number of those clash queries is reduced to 281. The expansion of the remaining 289 clash queries results in 44,072 queries that have to be evaluated within the generation of explanations. Note that we do not consider clash queries of type (iii) in our evaluation, since they will produce only singleton MISAs (resulting from clash type (c): incorrect datatypes) whose resolution is trivial and not crucial in federated settings.

We have run our implementation of detecting and repairing inconsistency, called ClashSniffer, on a CentOS 6.7 virtual machine consisting of 4x Intel Xeon CPUs (à 4 cores @ 2.50 GHz) and 128 GB of RAM. The Virtuoso instances are hosted in an Ubuntu 14.04 LTS virtual machine with 6x Intel Xeon CPUs (à 4 cores @ 2.60 GHz) and 96 GB of RAM (16 GB of RAM are assigned to each Virtuoso instance). The runtime for inconsistency detection and the generation for appropriate explanations over all four data sources takes 80.1 min (minutes), where 56.5 min are required for evaluating the query parts. This runtime depends on the performance of the machines that host the data sources. The runtime for repair generation takes 6 min for the first phase and 12.2 min for the second phase.

Table 1 summarizes the characteristics of each data source and depicts the results of our experimental evaluation. Beside showing statistics for each data source on its own, the table shows two federated settings on which we have run our implementation of detecting and repairing inconsistency. We have defined the first federated setting $\mathcal{F}$ that comprises data source $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, whereas the second one, referred to as $\mathcal{F}'$, comprises all four data sources. The runtimes presented in the previous paragraph refer to the $\mathcal{F}'$ setting, which comprises more than 284 million triples. We analyze these two settings in order to understand the impact of adding an additional data source, since we expect that the availability of an additional data source comprising complementary and potentially redundant information should have a positive impact not only on the quality of the repair but also on the quantity of MISAs solved by the repair. Note that we compare the two federated settings also against the local settings, where we apply the approach to each data source on its own. For that reason we have also added a row to the table headed with the Σ symbol, where we sum up the numbers for all single data sources.

Table 1. Results of MISAs and Repairs

	♯triples	♯$\mathcal{C}$ -MISAs-	$\lvert\mathcal{R}\rvert$ -resolvable repair-	$\lvert\mathcal{R}'\rvert$ -learned repair-	rem. MISA rate
$\mathcal{A}_1$	72,372,256	3,266,765	46,128 (291,025)	1,187,461 (1,188,115)	54.72%
$\mathcal{A}_2$	17,765,873	1,096,337	4,654 (15,525)	246,289 (247,180)	76.0%
$\mathcal{A}_3$	166,320,474	12,016,391	1,024,414 (2,057,807)	420,081 (433,827)	79.26%
$\mathcal{A}_4$	27,897,291	26,504	521 (23,419)	4 (4)	11.62%
Σ	284,355,894	16,405,997	1,075,717 (2,387,776)	1,853,835 (1,869,126)	74.05%
$\mathcal{F}$	256,458,603	16,605,398	1,109,524 (4,770,557)	3,971,584 (10,368,294)	8.83%
$\mathcal{F}'$	284,355,894	18,146,950	1,993,136 (7,166,005)	3,267,659 (9,574,136)	7.75%

The first data column of Table 1 illustrates the size of each data source and each federated KB, respectively. The second column depicts the number of detected clashes. The largest data source is ePrints Open Archives ($\mathcal{A}_3$) with more than 160 million triples and is also the data source with the highest number of local clashes. While local clashes are dominant in the dataset, we can also see that more conflicts can be detected by analyzing the data sources in a federated setting. The numbers increase from ≈16.4 (Σ) to ≈16.6 ($\mathcal{F}$) and to ≈18.1 million clashes ($\mathcal{F}'$). The clashes detected in $\mathcal{F}'$ comprise 12,209,235 clashes (67.3%) that result from functionality assertions where 0.5% of these are federated and 5,937,715 clashes (32.7%) caused by negative inclusions with a rate of 28.3% federated clashes.

Figure 1 shows the number of all federated clashes (MISAs) and how they are distributed on the pairs of data sources in setting $\mathcal{F}'$. Note that each possible combination of data sources results in more than 40,000 clashes. Despite the fact that both data source $\mathcal{A}_1$ and $\mathcal{A}_4$ are based upon DBLP, it is interesting to see that these two data sources produce 77% of all federated clashes. A reason for this could be that the underlying DBLP dataset is parsed, converted, and interpreted differently and is mapped to distinct TBoxes.

The numbers in column three and four of Table 1 show the number of resolvable and learned repairs that are generated by our algorithm. The values in parenthesis represent the numbers of MISAs that are resolved by the generated repair. Note that this number is often significantly higher than the size of the repair, which indicates a high overlap of the MISAs. The last column comprises the rate of remaining clashes after our algorithm was applied. It is interesting to see that the rate of remaining MISAs in the federated settings $\mathcal{F}$ and $\mathcal{F}'$

[3] http://dblp.l3s.de
[4] http://www.bibsonomy.org
[5] http://foreign.rkbexplorer.com
[6] http://dblp.rkbexplorer.com
[7] http://www.researchgate.net/publication/299852903

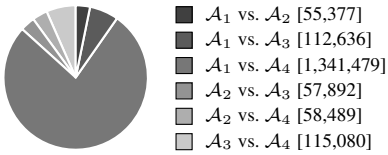


Figure 1. Distribution of Federated MISAs

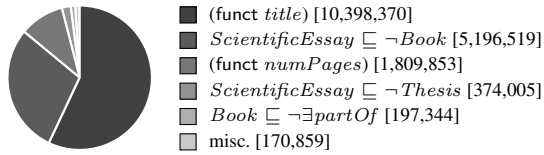


Figure 2. Axioms Causing Inconsistency

is significantly lower than applying our algorithm in a local setting of each single data source. Adding the additional data source $\mathcal{A}_4$ ($\mathcal{F}$ vs. $\mathcal{F}'$) results in a larger size of the resolvable repair but in a lower size of the learned repair. Moreover, this results in a higher number of new conflicts, but decreases also the relative number of remaining MISAs. Another positive effect is, that MISAs that are not resolvable in the first phase of $\mathcal{F}$ are now solved in the first phase of $\mathcal{F}'$, why the number of the learned repair is decreased in $\mathcal{F}'$.

Table 2 highlights the data source specific impact of our approach. We compare the size of the local repair for each $\mathcal{A}_i$, taking only inconsistencies caused by assertions from $\mathcal{A}_i$ into account, against the size of the subset of the federated repair restricted to assertions from $\mathcal{A}_i$. We conducted the comparison for both resolvable and learned repair of each federated setting $\mathcal{F}$ and $\mathcal{F}'$. We can see, for example, that in the federated setting $\mathcal{F}$ the size of the resolvable repair for $\mathcal{A}_1$ was increased by 13.59% and that the learned repair for $\mathcal{A}_1$ was increased by 66.95%. For this data source the effect of the additional data source in $\mathcal{F}'$ is evidenced, since the size of the resolvable repair for $\mathcal{A}_1$ was increased by $1,896.71\%$ compared to the local application of our approach at this data source. The reason for this significant increase is not only the fact that more MISAs of data source $\mathcal{A}_1$ can be solved (represented by the decreased rate of remaining MISAs), but also due to the decrease of the learned repair by -70.61%. Hence, more MISAs of this data source become resolvable, while in the federated setting $\mathcal{F}$ these MISAs can be resolve only by the learned repair. This positive impact on the quality of the repair and also on the quantity of MISAs solved by the repair by an additional data source, comprising complementary and potentially redundant information, is also reflected by data source $\mathcal{A}_2$. The last row shows the values based on summing up the results for all data sources. In average $+84.04\%$ are gained for the resolvable repair and this has again an impact measured in terms of $+68.55\%$ for the learned repair in $\mathcal{F}'$. Note that the federated setting has also an impact on reducing the number of local MISAs due to the fact that assertions from other data sources interfere with the assertions from local MISAs. Overall, the rate of remaining MISAs can be significantly reduced from originally 74.05% in local application (see Table 1) to 8.57% in federated setting $\mathcal{F}'$. These numbers illustrate the positive impact of the federated setting which allows to achieve a significantly higher recall rate for detecting problematic assertions.

Table 2. Impact of Federated Debugging

	$\mathcal{F}$			$\mathcal{F}'$		
	$\Delta\mathcal{R}$ **resolvable repair**	$\Delta\mathcal{R}'$ **learned repair**	**rem. MISA rate**	$\Delta\mathcal{R}$ **resolvable repair**	$\Delta\mathcal{R}'$ **learned repair**	**rem. MISA rate**
$\mathcal{A}_1$	+13.59%	+66.95%	24.37%	+1,896.71%	-70.61%	22.81%
$\mathcal{A}_2$	+236.53%	+3.1%	59.49%	+287.22%	+2.18%	59.49%
$\mathcal{A}_3$	+1.51%	+498.46%	0.15%	+1.52%	+500.83%	0.05%
$\mathcal{A}_4$	–	–	–	+37.24%	-100.0%	11.62%
Σ	+3.04%	+156.25%	8.95%	+84.04%	+68.55%	8.57%

To give an insight into the generated MISAs as well as the resolvable and learned repair for federated setting $\mathcal{F}'$ we have done some further analysis. Starting with the generated MISAs, the axioms causing inconsistency are depicted in Figure 2. As already mentioned, most of the clashes result from functionality assertions, especially for the attribute $title$. Most of the clashes that are caused by a negative inclusion result from the axiom $ScientificEssay \sqsubseteq \neg Book$.

The computation of resolvable repair in the federated setting $\mathcal{F}'$ comprised 413 iterations of the while loop in Algorithm 1. The highest cardinality found was 18,189. The reason for this high cardinality is that "*Bioinformatics*"[8], which is a journal series comprising lots of articles assigned in data source $\mathcal{A}_1$ and $\mathcal{A}_4$, is wrongly defined as an article in data source $\mathcal{A}_3$.

After generating the resolvable repair the trust values for all concepts, roles and attributes occurring in unresolved MISAs are calculated with respect to each data source. The top 5 of lowest trust values derived are depicted in Table 3. Especially the two lowest trust values lead to the conclusion that assertions on $volume$ attributes are probably misused in $\mathcal{A}_1$ and $\mathcal{A}_3$. Having a more detailed look into the datasets of those sources confirms this conclusion, since $volume$ attributes are in both data sources not used at the level of collections like proceedings, journals or books, but on the level of articles published in a collection. The low trust values for $volume$ attributes reflect also the fact that the negative inclusion $ScientificEssay \sqsubseteq \neg Book$ is part of the top 5 axioms causing inconsistency (see Figure 2), since the expansion of $Book$ comprise $\exists volume$.

Table 3. Top 5 of Lowest Trust Values

trust value	**data source**	$\sigma \in \Sigma$
0.1045	$\mathcal{A}_3$	*http://purl.org/ontology/bibo/volume*
0.2741	$\mathcal{A}_1$	*http://swrc.ontoware.org/ontology♯volume*
0.8889	$\mathcal{A}_1$	*http://swrc.ontoware.org/ontology♯MasterThesis*
0.9476	$\mathcal{A}_2$	*http://swrc.ontoware.org/ontology♯Booklet*
0.9587	$\mathcal{A}_2$	*http://swrc.ontoware.org/ontology♯Unpublished*

Finally, we have analyzed the remaining MISAs that are not solved by our approach. All of them are not federated and are exclusively caused by functionality assertions. Approximately, 53% of the remaining MISAs comprise the axiom (funct $numPages$) and 47% the axiom (funct $title$).

To evaluate the quality of the generated repairs 100 randomly selected MISAs of each phase in each federated setting are manually evaluated by three persons. Table 4 presents the precision of our debugging approach based on the sample we analyzed. If an URI is not accessible or at least two persons did not come to the same decision, the decision specific case is annotated as uncertain.

The evaluation results indicate a high precision of our approach and substantiate that the removal decisions are based on a reason-

[8] http://bioinformatics.oxfordjournals.org/

Table 4. Quality of Repairs

		correct	incorrect	uncertain
resolvable repair	$\mathcal{R}_{\mathcal{F}}$	94%	0%	6%
	$\mathcal{R}_{\mathcal{F}'}$	97%	0%	3%
learned repair	$\mathcal{R}'_{\mathcal{F}}$	96%	0%	4%
	$\mathcal{R}'_{\mathcal{F}'}$	84%	2%	14%

able heuristics. The measured precision scores also confirm that the majority voting scheme underlying our approach is a valid premise which ensures also a high precision of the second phase, where we use the statistical evidence of the first phase to apply the notion of trust to the remaining MISAs. This is also suggested by the fact that the precision scores for resolvable and learned repairs are roughly in the same range. However, the subsets we analyzed are not comprehensive enough to enable a conclusion about the impact of adding an additional data source. The precision of the resolvable repair is slightly increased, at the same time we observe a marginal drop for the learned repair. While we cannot prove a positive impact on precision, the previously presented results in terms of detected MISAs and repair sizes have clearly shown the positive impact on recall.

6 Related Work

State-of-the-art DL reasoners that are used for inconsistency detection and its explanations basically process local KBs and are therefore inappropriate for distributed environments. Moreover, regardless of the supported language expressiveness or the underlying reasoning method (such as widely used tableau algorithms as in FaCT++ [23] or Pellet [22]; the hypertableau technique of HermiT [9]; consequence-driven approaches like the ones described by Kazakov [13]; and resolution based methods described by Motik & Sattler [16]), they do not deal with inconsistency detection in a federated setting.

To the best of our knowledge there is currently no ready to use approach that addresses inconsistency detection and generation of repairs in the context of federated KBs. However, there are some works in a similar direction.

Bonatti et al. [3] proposed an approach that can be applied to a scenario similar to the one we analyzed in our experiments. Their approach is based on annotated logic programs for tracking indicators of provenance and trust during the reasoning process. However, the reasoning itself is not conducted in a given federated setting, while our approach works directly on top of existing SPARQL interfaces. This shall not be confused with the fact that Bonatti et al. described a distributed implementation of the algorithm. Another important difference is the origin of the trust values. While we derive the trust values required for the second phase from explicit and implicit conflicts in the data sources using reasoning in the first phase, Bonatti et al. apply a well-known page rank algorithm that does not at all consider logical dependencies.

Calvanese et al. [4] present apart from the initial definition of the *DL-Lite* family among others a definition of a translation function δ that transforms negative inclusions and functionality assertions into queries (FOL formulas). This translation function is applied in the algorithm Consistent to each negative inclusion and functionality assertion that can be logically implied from the given TBox. Afterwards, a Boolean query comprising the union of all queries generated by δ is evaluated over the given ABox. In contrast to our approach the work of Calvanese et al. do not support *DL-Lite*$_{\mathcal{A}}$ KBs. Besides this, Calvanese et al. [5] expand their approach to *DLR-Lite*$_{\mathcal{A},\sqcap}$, a new member of the *DL-Lite* family that permits among others the use of *n*-ary relations and conjunctions on the left-hand side of inclusion assertions. Despite the fact that the algorithms Consistent proposed in these works are similar to our approach, both only identify if there is an inconsistency but do not specify these inconsistencies in greater detail or give some explanations to them. Furthermore, our approach additionally comprises the federation of distributed *DL-Lite* KBs.

The approach proposed by Lembo et al. [14, 15] facilitate meaningful query results over inconsistent *DL-Lite* KBs under different inconsistency-tolerant semantics. Therefore, an additional rewriting under the defined semantics is applied to the rewritings produced by PerfectRef in order to implement inconsistency tolerance on query answering. Roughly speaking, queries generated by applying backward-chaining are extended such that triples producing inconsistency will not be considered on query answering. For this purpose, similar to our approach ontology axioms that can be contradicted by ABox assertions are used for query generation, i.e., its expansion, but with the difference that their aim is to exclude all assertions that cause inconsistency from query evaluation whereas our claim is to select these assertions, which is exactly the opposite. Although the method of Lembo et al. is suitable for accessing distributed data, it is not designed for inconsistency detection and explanation.

Several approaches have already been proposed to solve the problem of repair plan generation [8]. Depending on the specifics of the setting one might, e.g., be interested to remove a minimum number of assertions causing inconsistency by computing a smallest minimal hitting set over all explanations. However, to the best of our knowledge, none of these approaches consider federated settings and make precise distinctions between assertions occurring in different data sources.

7 Conclusions

In this paper we have described an approach for detecting and resolving inconsistency in federated large scale KBs. The approach is based on the generation of clash queries which are known to be complete for inconsistency detection in *DL-Lite*$_{\mathcal{A}}$ KBs. Once all logical conflicts have been collected, a majority voting scheme is applied in the first phase to determine a partial repair. This approach does not aim at generating a global optimal repair, but applies an efficient heuristic where each step in the algorithm corresponds to a reasonable decision. Based on determining the degree of trust for each assertion type with respect to each data source by analyzing the partial repair, we are able to extend the repair in the second phase.

We applied the approach in a federated setting using four LOD data sources from the domain of library science. Overall, the federated KB consists of more than 284 million triples and we are able to detect 18.1 million conflicts. The results of our experiments show that we are able to solve 92.25% of those conflicts by the proposed approach. Furthermore, according to our evaluation, we can conclude that the rate of remaining MISAs can be reduced by taking the federated setting into account. By manually annotating samples from the generated repairs, we measured a precision between 84% and 97%, which is a surprisingly good result for a fully automated approach.

So far we have focused on ABox assertions of a federated KB. In our future work, we will address a combined approach in which we try to strike a balance between repairing a potentially erroneous TBox and clashing ABox assertions. Furthermore, the assessment of trustworthiness in `owl:sameAs` statements is an open issue that will also be addressed in our future work.

REFERENCES

[1] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev, 'The dl-lite family and relations', *Journal of artificial intelligence research*, **36**(1), 1–69, (2009).

[2] Franz Baader, *The description logic handbook: theory, implementation, and applications*, Cambridge: Cambridge University Press, 2003.

[3] Piero A Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro, 'Robust and scalable linked data reasoning incorporating provenance and trust annotations', *Web Semantics: Science, Services and Agents on the World Wide Web*, **9**(2), 165–201, (2011).

[4] Diego Calvanese, Giuseppe De Giacomo, 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, (2007).

[5] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati, 'Data complexity of query answering in description logics', *Artificial Intelligence*, **195**, 335–360, (2013).

[6] Diego Calvanese, Martin Giese, Dag Hovland, and Martin Rezk, 'Ontology-based integration of cross-linked datasets', in *The Semantic Web - ISWC 2015*, 199–216, Springer, (2015).

[7] Giorgos Flouris, Zhisheng Huang, Jeff Z Pan, Dimitris Plexousakis, and Holger Wache, 'Inconsistencies, negations and changes in ontologies', *Proceedings of the National Conference on Artificial Intelligence*, **21**(2), 1295, (2006).

[8] Peter Haase and Guilin Qi, 'An analysis of approaches to resolving inconsistencies in DL-based ontologies', in *Proceedings of the International Workshop on Ontology Dynamics (IWOD-07)*, pp. 97–109, (2007).

[9] Ian Horrocks, Boris Motik, and Zhe Wang, 'The HermiT OWL Reasoner', in *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), Manchester, UK*, (2012).

[10] Qiu Ji, Peter Haase, Guilin Qi, Pascal Hitzler, and Steffen Stadtmüller, 'Radonrepair and diagnosis in ontology networks', in *The semantic web: research and applications*, 863–867, Springer, (2009).

[11] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin, 'Finding all justifications of OWL DL entailments', in *The Semantic Web*, pp. 267–280. Springer, (2007).

[12] Richard M. Karp, 'Reducibility among combinatorial problems', in *Proceedings of a symposium on the Complexity of Computer Computations*, pp. 85–103, (1972).

[13] Yevgeny Kazakov, 'Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies.', in *IJCAI*, volume 9, pp. 2040–2045, (2009).

[14] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo, 'Query rewriting for inconsistent DL-Lite ontologies', in *Web Reasoning and Rule Systems*, 155–169, Springer, (2011).

[15] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo, 'Inconsistency-Tolerant First-Order Rewritability of DL-Lite with Identification and Denial Assertions', in *Proceedings of the 25th International Workshop on Description Logics*, (2012).

[16] Boris Motik and Ulrike Sattler, 'A comparison of reasoning techniques for querying large description logic aboxes', in *Logic for programming, artificial intelligence, and reasoning*, pp. 227–241. Springer, (2006).

[17] Andreas Nolle, Christian Meilicke, Heiner Stuckenschmidt, and German Nemirovski, 'Efficient federated debugging of lightweight ontologies', in *Web Reasoning and Rule Systems*, 206–215, Springer, (2014).

[18] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati, 'Linking data to ontologies', in *Journal on data semantics X*, 133–173, Springer, (2008).

[19] Guilin Qi, Qiu Ji, and Peter Haase, 'A conflict-based operator for mapping revision', in *The Semantic Web-ISWC 2009*, 521–536, Springer, (2009).

[20] Raymond Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**, 57–95, (1987).

[21] Sebastian Rudolph, 'Foundations of description logics', in *Reasoning Web. Semantic Technologies for the Web of Data*, 76–136, Springer, (2011).

[22] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz, 'Pellet: A practical owl-dl reasoner', *Web Semantics: science, services and agents on the World Wide Web*, **5**(2), 51–53, (2007).

[23] Dmitry Tsarkov and Ian Horrocks, 'FaCT++ description logic reasoner: System description', in *Automated reasoning*, 292–297, Springer, (2006).

[24] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner, 'Ontology-based integration of information - a survey of existing approaches', in *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pp. 108–117. Citeseer, (2001).