Belief Base Revision for Datalog+/- Ontologies

Songxin Wang[1,2], Jeff Z. Pan[2], Yuting Zhao[2], Wei Li[3], Songqiao Han[1], and Dongmei Han[1]

[1] Department of Computer Science and Technology
Shanghai University of Finance and Economics, China
[2] Department of Computer Science, University of Aberdeen, UK
[3] School of Computer Science, Fudan University, China

Abstract. Datalog+/- is a family of emerging ontology languages that can be used for representing and reasoning over lightweight ontologies in Semantic Web. In this paper, we propose an approach to performing belief base revision for Datalog+/- ontologies. We define a kernel based belief revision operator for Datalog+/- and study its properties using extended postulates, as well as an algorithm to revise Datalog+/- ontologies. Finally, we give the complexity results by showing that query answering for a revised linear Datalog+/- ontology is tractable.

Keywords: Datalog+/-, ontology, belief revision, kernel

1 Introduction

Datalog+/- [3] has its origin from database technologies, encompasses and generalizes the tractable description logics EL and DL-Lite, which can be used to represent lightweight ontologies in Semantic Web [6]. Datalog+/- enables a modular rule-based style of knowledge representation. Its properties of decidability of query answering and good query answering complexity in the data complexity allows to realistically assume that the database D is the only really large object in the input. These properties together with its expressive power make Datalog+/- a useful tool in modelling real applications such as ontology querying, web data extraction, data exchange, ontology-based data access and data integration.

Belief revision deals with the problem of adding new information to a knowledge base in a consistent way. Ontologies are not static, e.g., they may evolve over time, so it is important to study belief revision for Datalog+/- ontologies. For both classic logic and description logic, belief revision has been studied intensively and many literature exist, such as [1, 7]. There are some extensions of Datalog+/- to deal with incomplete or inconsistency information, including inconsistent handling method [4], probability extension [5, 8], and well-founded semantics extension [9], however, to the best of our knowledge, there is no belief revision method for Datalog+/- before.

In this paper, we address the problem of belief revision for Datalog+/- ontologies and propose a *kernel-incision* based belief revision operator. Kernel consolidation was originally introduced by Hansson [10] based on the notion of kernels and incision function. The idea is that, given a knowledge base KB that needs to be consolidated (*i.e.*, KB is inconsistent), the set of kernels is defined as the set of all minimal inconsistent

subsets of KB. For each kernel, a set of sentences is removed (i. e. , an incision is made) such that the remaining formulas in the kernel are consistent. Note that it is enough to remove any single formula from the kernel because they are minimal inconsistent sets. The result of consolidating KB is then the set of all formulas in KB that are not removed by the incision function.

We adopt the kernel-based consolidation idea into Datalog+/- ontologies, and give an approach to deal with belief revision in the face of new information which contains two parts of message: (i) the *new facts* A, and (ii) the *unwanted set* Ω. With our belief revision operator, a Datalog+/- ontology KB is able to be undated by adding new facts A in its database, and removing some database element to prevent any result in the unwanted set Ω, such as inconsistency $\bot$. We study the properties of proposed approach using extended postulates, and then give algorithms to revise the Datalog+/- ontologies. We finally give the complexity results by showing that query answering for a revised Datalog+/- ontology is tractable if a linear KB is considered.

The paper is organized as follows. In Section 2 we introduce some preliminary knowledge about Datalog+/-. In Section 3 we give our revision operator on Datalog+/- ontologies. The properties of the operator are investigated in Section 4. In Section 5 we give the algorithm and provide complexities results. Related works and the conclusion are given in Section 6 and 7 respectively.

2 Preliminaries on Datalog+/-

In this section, we briefly recall some necessary background knowledge on Datalog+/-.

Databases and Queries We assume (i) an infinite universe of *(data)constants* constants Δ (which constitute the normal domain of a database), (ii) an infinite set of *(labelled) nulls* Δ_N (used as fresh Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (unique name assumption), while different nulls may represent the same value. We also assume that there is a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, ..., X_n$ with $k \geq 0$.

We assume a relational schema $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula*(or *atom*) **a** has the form $P(t_1, ..., t_n)$, where P is an n-arry predicate, and $t_1, ..., t_n$ are terms.

We assume a *Database (instance) D* for $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query* (CQ) over R has the form $Q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu : \Delta \cup \Delta_N \cup \mathcal{V} \rightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$ and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms.

The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists\mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples t over Δ for which there exists a homomorphism $\mu : \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu : \Phi(X, Y) \subseteq D$ and $\mu(X) = t$. The *answers* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, *a tuple-generating dependency* (TGD) σ is a first-order formula of the form $\forall\mathbf{X}\forall\mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists\Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and *head* the head of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D , there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ.A TGD σ is *linear* iff it contains only a single atom in its body. *Query answering* under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of models of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. It is proved that query answering under general TGDs is undecidable, even when the schema and TGDs are fixed.

For a BCQ Q we say that $(D \cup \Sigma)$ *entail* Q if answer for a BCQ Q to D and Σ is *Yes*, $D \cup \Sigma$ *entail* a set of BCQs if it entail some element of it.

Negative constraints (NC): A negative constraints γ is a first-order formula $\forall\mathbf{X}\, \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the body of γ) is a conjunction of atoms (without nulls and not necessarily guarded).

Equality-generating dependencies (EGD): A equality-generating dependency σ is a first-order formula of the form $\forall\mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of Σ, denoted $body(\sigma)$, is a (without nulls and not necessarily guarded) conjunction of atoms, and X_i and X_j are variables from X . Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y}) \subseteq D$, it holds that $h(X_i) = h(X_j)$.

The Chase The *chase* was first introduced to enable checking implication of dependencies, and later also for checking query containment. It is a procedure for repairing a database relative to a set of dependencies, so that the result of the chase satisfies the dependencies. By *chase*, we refer both to the chase procedure and to its result.

The chase works on a database through TGD and EGD chase rules. Let D be a database, and Σ a TGD of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists\mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, Σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(X, Y)$ to atoms of D. Let Σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$ where z_j is a fresh null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* Σ on D adds to D the atom $h_1(\Psi(X, Z))$ if not already in D.

The chase algorithm for a database D and a set of TGDs consists of essentially an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for D and Σ.

The chase relative to TGDs is a *universal model*, that means, there exists a homomorphism from $chase(D, \Sigma)$ onto every $B \in mods(D, \Sigma)$. This implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $chase(D, \Sigma) \models Q$. For guarded TGDs Σ, such BCQs Q can be evaluated on an initial fragment of $chase(D, \Sigma)$ of constant depth k$|Q|$, which is possible in polynomial time in the data complexity.

Datalog+/- ontology A *Datalog+/- ontology* KB=(D, Σ), where D is database, $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}. We say KB is *linear* iff Σ_T is linear. A Datalog+/- ontology KB=(D, Σ) is *consistent*, iff $model(D, \Sigma) \neq \emptyset$, otherwise it is inconsistent.

3 Belief base revision in Datalog+/- ontology

We give an approach for performing belief base revision for Datalog+/- ontology in this section. We present a framework based on kernels and incision functions to deal with the problem of belief revision for Datalog+/- ontologies.

3.1 Revision

Firstly we give the definition of kernel in Datalog+/- ontologies.

Definition 1 (Kernel). *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω. A kernel is a set $c \subseteq D \cup A$ such that (c, Σ) entail Ω, and there is no $c' \subset c$ such that (c', Σ) entail Ω. We denote $(D \cup A) \perp\!\!\!\perp \Omega$ the set of all kernels.*

Example 1. A (guarded) Datalog+/- ontology KB=(D, Σ) is given below. Here, the formulas in Σ_T are tuple-generating dependencies (TGDs), which say that each person working for a department is an employee (σ_1), each person that directs a department is an employee (σ_2), and that each person that directs a department and works in that department is a manager (σ_3). The formulas in Σ_{NC} are negative constraints, which say that if X supervises Y, then Y cannot be a manager (ν_1), and that if Y is supervised by someone in a department, then Y cannot direct that department (ν_2). The formula Σ_E is an equality-generating dependency (EGD), saying that the same person cannot direct two different departments.

D=$\{directs(tom, d_1), directs(tom, d_2), worksin(john, d_1), worksin(tom, d_1)\}$;
Σ_T=$\{\sigma_1 : worksin(X, D) \rightarrow emp(X), \sigma_2 : directs(X, D) \rightarrow emp(X),$
$\sigma_3 : directs(X, D) \wedge worksin(X, D) \rightarrow Manager(X)\}$;

Σ_{NC}={$\nu_1 : supervises(X, Y) \wedge manager(Y) \rightarrow \bot$,
$\nu_2 : supervises(X, Y) \wedge worksin(X, D) \wedge directs(Y, D) \rightarrow \bot$};
Σ_E={$\nu_3 : directs(X, D_1) \wedge directs(X, D_2) \rightarrow D_1 = D_2$}.

If new observation is supervises(tom, john), unwanted sentence is emp(tom), then kernel is $c_1 = \{worksin(tom, d_1)\}$.

If new observation is supervises(tom, john), unwanted atom is $\bot$, then kernels are
$c_1 = \{supervises(tom, john), direct(tom, d_1), worksin(john, d_1)\}$
$c_2 = \{supervises(tom, john), direct(tom, d_1), worksin(tom, d_1)\}$
$c_3 = \{direct(tom, d_1), direct(tom, d_2)\}$.

We then give the definition of incision function in Datalog+/- ontologies.

Definition 2 (Incision Function). *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, an incision function is a function σ that satisfies the following two properties*

1. $\sigma((D \cup A) \perp\!\!\!\perp \Omega) \subseteq \bigcup((D \cup A) \perp\!\!\!\perp \Omega)$.
2. *if* $X \in ((D \cup A) \perp\!\!\!\perp \Omega)$ *then* $X \cap \sigma((D \cup A) \perp\!\!\!\perp \Omega) \neq \emptyset$.

We now give the definition of revision operator. The idea is to add the new information to ontology and then cut unwanted information from it in a consistent way. Intuitively, the revision intends to add A and avoid Ω in a rational way. Note that we cut unwanted database Ω, which is a generation of just cutting $\bot$ from the ontology to avoid the contravention. In this point we are like Ribeiro *et al.*[11].

Definition 3 (Revision Operator). *Given a Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, let $D' = (D \cup A) \backslash \sigma((D \cup A) \perp\!\!\!\perp \Omega)$, where σ is an incision function, then revised ontology $KB * A = (D', \Sigma)$.*

Example 2. We continue with example 1. If new observation is supervises(tom, john), unwanted atom is emp(tom), let incision function be {$worksin(tom, d1)$}, then revised database is
$D' = \{directs(tom, d_1), directs(tom, d_2), worksin(john, d_1)\}$.

If new observation is supervises(tom, john), unwanted atom is $\bot$, let incision function be $\{supervises(tom, john), direct(tom, d_2)\}$, then revised database is
$D' = \{directs(tom, d_1), worksin(john, d_1), worksin(tom, d_1)\}$.

4 General Properties

It is important to ensure the revision operator behave rationally, so we analyze some general properties of it in this section. We first propose the revision postulates for Datalog+/- ontologies, which is an adaptation of known postulates for semi-revision, then prove that these new postulates are indeed satisfied by the operator.

Definition 4 (Postulates). *Given a Datalog+/- ontology $KB = (D, \Sigma)$, a new observation database instance A, an unwanted database instance Ω, let $KB \diamond A = (D', \Sigma)$ be the revised ontology, then the postulates are:*

1. *(Consistency) Any element of* Ω *is not entailed by* $KB\diamond A$.
2. *(Inclusion))* $D' \subseteq D \cup A$.
3. *(Core-retainment) if* $\beta \in D$ *and* $\beta \notin D'$*, then there is* D'' *such that* $D'' \subseteq D \cup A$, $(D'' \cup \{\beta\}, \Sigma)$ *entail* Ω*, but* (D'', Σ) *does not entail* Ω.
4. *(Internal change) If* $A \subseteq D, B \subseteq D$ *then* $KB\diamond A = KB\diamond B$.
5. *(Pre-expansion)* $(D \cup A, \Sigma)\diamond A = KB\diamond A$.

The intuitive meaning of the postulates are as follows: *consistency* means no atom in the unwanted database should be entailed; *inclusion* means that no new atoms were added; *core-retainment* means if an atom is deleted, then there must be some reason; *internal change* means that every time an ontology is revised by any of its own elements of database, then the result ontology should be same; *pre-expansion* means that if an ontology is expanded by a database and then revised by it, the result should be the same as revising the original ontology by the database.

We now prove that the revision operator given in the last section satisfy these postulates.

Theorem 1. *Given a Datalog+/- ontology* $KB = (D, \Sigma)$*, new observation database instance* A*, unwanted database instance* Ω*, let* $KB * A = (D', \Sigma)$ *be the revised ontology defined as above, then the revision satisfies the postulates in Def. 4.*

Proof. Inclusion, internal change, pre-expansion follows directly from the construction. To prove *consistency*, assume by contradiction that it is not. Then there is an element of Ω that is entailed by $KB\diamond A$, as Datalog+/- is a fragments of first-order logic, from compacity of first-order logic it follows that there is a $Z \in D \cup A$, such that there is an element of Ω that is entailed by (Z, Σ). We can then infer by monotonicity that there is a $Z' \subseteq Z$ such that $Z' \in (D \cup A) \perp\!\!\!\perp \Omega$. Then we must have $Z' \neq \emptyset$, and by construction there must be $\epsilon \in \sigma((D \cup A)) \cap Z'$, but if this is true then $\epsilon \notin (D \cup A)$ and $\epsilon \in Z' \subseteq (D \cup A)$, which is a contradiction. To prove *core-retainment*, we have that if $\beta \in D$ and $\beta \notin D'$, then there is $\beta \in \sigma((D \cup A) \perp\!\!\!\perp \Omega)$, That is, there is a $X \in (D \cup A) \perp\!\!\!\perp \Omega$ such that $\beta \in X$. Considering $D'' = X/\beta$, then $D'' \subseteq D \cup A$, $(D'' \cup \{\beta\}, \Sigma)$ entail Ω, but (D'', Σ) does not entail Ω.

5 Algorithms

In the section, we first give an algorithm to compute all kernels when an atom is unwanted, and then give the algorithm of revision. We deal with *linear* ontology in this section.

5.1 Computing kernels for an atom

Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted atom ω, we give in this subsection a method to calculate all kernels $(D \cup A) \perp\!\!\!\perp \omega$.

We first give the method when unwanted atom λ is a node in *Chase* of $(D \cup A, \Sigma)$. The idea is to travel chase graph bottom-up and then cut non-minimal ones from result

sets of atoms. Note that this idea is similar to the one used in Lukasiewicz etc [4], which use a bottom-up travel of chase graph starting from the matching a body every rule of Σ_{NC} to compute the culprits of an inconsistent ontology.

We now give algorithm Justs(λ). Min(Justs(λ)) are exactly kernels $(D \cup A) \perp\!\!\!\perp \lambda$, where Min is used in the usual sense of subset inclusion. Note that *just* below is not a set of nodes, but a set of sets of nodes. Note also that step 2 can be done because we can collect all information needed from *Chase*.

Algorithm 1 Justs(λ)

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation database instance A, and a node λ in *Chase* of $(D \cup A, \Sigma)$.
Ensure: Justs(λ)
1: $just = \emptyset$
2: **for all** $\Phi_i \subseteq$ (nodes in *Chase*) such that there is rule $r : \Phi(X, Y) \rightarrow \exists Z\ \Psi(X, Z)$ which is applicable to Φ_i and produces λ **do**
3: $just = just \cup \{\Phi_i\}$
4: **end for**
5: **for all** $\Phi_i \in just$ **do**
6: **for all** $\Phi_i^j \in \Phi_i$ **do**
7: **if** Justs(Φ_i^j) $\neq \emptyset$ **then**
8: $just$ = Expand($just$, Φ_i^j)
9: **end if**
10: **end for**
11: **end for**
12: **return** $just$

13: Expand($just$, a)
14: **for all** $\phi \in just$ **do**
15: **if** $a \in \phi$ **then**
16: $just = just/\phi$
17: **for all** $j\_a \in$ Justs(a) **do**
18: $just = just \cup \{\phi/a \cup j\_a\}$
19: **end for**
20: **end if**
21: **end for**
22: **return** $just$

We now have the following theorem.

Theorem 2. *Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, Let λ be a node in chase graph of Datalog+/- ontology $(D \cup A, \Sigma)$, then Min(Justs(λ))=$(D \cup A) \perp\!\!\!\perp \lambda$. The algorithm run in polynomial time in the data complexity.*

Proof. We first prove that $(Min(Justs(\lambda)), \Sigma)$ entail λ, that is, for every model M of $(Min(Justs(\lambda)), \Sigma)$, there is a homomorphism μ such that $\mu(\lambda) \subseteq M$.

In fact, we will show that for all nodes that were produced in the bottom-up travel process, it holds that the node is entailed by $(Min(Justs(\lambda)), \Sigma)$, Then, as a result, the λ is also entailed by $(Min(Justs(\lambda)), \Sigma)$ automatically.

Suppose the derivation level of λ is N. We prove level by level from 1 to N. If the node level is 1. let M be a model of $(Min(Justs(\lambda)), \Sigma)$, then $M \supseteq Min(Justs(\lambda))$, but from the definition of satisfaction of a rule, whenever there is a homomorphism that map the body to the database, there is a extended homomorphism that map the head to the database,so, this homomorphism map the node to M, the entailment holds.

Suppose for all node whose derivation level is n, it is right. That is, there is a a homomorphism that map the node to M, now we consider the node whose derivation level is n+1. Consider the rule that applicable and can get this node, since all parent nodes of the node has a level smaller or equal to n, so there is a homomorphism that map these nodes to M, as the rule itself is satisfied by M, we can then construct a new homomorphism by extend the above homomorphism to map the node to M. So we have $(Min(Justs(\lambda)), \Sigma)$ entail the node.

We now show that there are no other subsets of $D \cup A$ that along with Σ entail λ and is smaller than Min(Justs(λ)). Otherwise, suppose it is not, that is, there is a subset of $D \cup A$ that is smaller than Min(Justs(λ)) and along with Σ entail λ. Then we have a *ChaseGraph* that end with λ, however, according to the construction of the Min(Justs(λ)), this set should be equal to some element of Min(Justs(λ)), this is a contraction.

We finally shows that computing $Min(Justs(\lambda))$ in the linear case can be done in polynomial time in the data complexity. Note that the *ChaseGraph* is constant-depth and polynomial-size for a linear ontology due to the result in [3]. Note also that Justs(λ) is a recursive procedure, it will be called $N \times M$ times at most, where N is the depth of the graph and M is the numbers of rules in the Σ, and that at each time the algorithm is running, the time complexity exclusive the recursive procedure is polynomial, so the algorithm Min(Justs(λ)) run in polynomial time.

We now give the algorithm Kernels(ω) to compute kernels for an atom ω. Note that by match(ω, *Chase*) we mean the procedure of finding the same node as ω in *Chase*, if it is successful, return this node, otherwise return $\emptyset$.

Algorithm 2 Kernels(ω)

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation A, and an atom ω
Ensure: Kernels(ω)

```
compute the Chase of KB = (D ∪ A, Σ)
L=match(ω, Chase)
if L=∅ then
    return ∅
else
    return Min(Justs(L))
end if
```

Theorem 3 (Correctness and Complexity of Kernels(ω)). *Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation A, an unwanted atom ω, algorithm Kernels(ω) compute $(D \cup A) \perp\!\!\!\perp \omega$ correctly in polynomial time in the data complexity.*

Proof. If L=$\emptyset$, then $KB' = (D \cup A, \Sigma)$ do not entail the atom as *ChaseGraph* is sound and complete with respect to query answering. If L$\neq \emptyset$ then the atom is entailed by KB', in this case, MinJust(L) are all minimal sets of atoms that belong to database $D \cup A$ and along Σ entail ω according to theorem 1. So Kernels(ω) compute $(D \cup A) \perp\!\!\!\perp \omega$ correctly in both cases.

The complexity of the algorithm depends on the *match* procedure, as the *ChaseGraph* is polynomial-size and constant-depth for a linear ontology, the travel of *ChaseGraph* can be done in polynomial time, so the Kernels(ω) can be done in polynomial time.

5.2 Revision

We now give the algorithm to revise a linear Datalog+/- ontology named as *RevisionKB*.

Algorithm 3 RevisionKB

Require: a linear Datalog+/- ontology $KB = (D, \Sigma)$, a new observation A, unwanted instance Ω
Ensure: Revised ontology $KB * A$
1: **for** every atom ω, $\omega \in \Omega$ **do**
2: compute Kernels(ω)
3: **end for**
4: get all combinations of Kernels(ω, every combination corresponds to a way of choosing each element from Kernels(ω), where $\omega \in \Omega$.
5: $(D \cup A) \perp\!\!\!\perp \Omega$=minimal subsets of all combinations
6: $KB * A$=$((D \cup A) \backslash \sigma((D \cup A) \perp\!\!\!\perp \Omega), \Sigma)$

Note that $((D \cup A), \Sigma)$ may be an inconsistent ontology, however, inconsistency can be removed in the revised ontology by making $\perp \subseteq \Omega$.

We now show that the algorithm can compute revision of ontology and give the complexity.

Theorem 4 (Correctness and Complexity of RevisionKB). *Given a linear Datalog+/- ontology $KB = (D, \Sigma)$, new observation database instance A, unwanted database instance Ω, algorithm RevisionKB can compute revision correctly in polynomial time in the data complexity.*

Proof. Note that $(D \cup A) \perp\!\!\!\perp \Omega$ can be obtained by combining $(D \cup A) \perp\!\!\!\perp \omega$ for every elements $\omega \in \Omega$ and cut from it the non-minimal ones, the algorithm's correctness then follows directly from the definition of revision operator.

The complexity of the algorithm depends basically on the task of finding Kernels(ω), as it run in polynomial time due to Theorem 3, so we have the conclusion.

6 Related Works

There is strong relationship between Datalog+/- ontology and description logic as they can be translated to each other in many cases. In the area of belief revision in description logic, Ribeiro *et al.* [11] bear much similarities to our work since they also used a kernels-based semi-revision method, they give a belief revision method to a monotonic logic, take description logic as a spacial case. However, there are difference between their work and this paper. They deal with monotonic logic, but Datalog+/- has a different syntax and semantic and cannot been cover by it. Furthermore, Ribeiro *et al.* [11] compute kernels by invoke classical reasoning, but this paper give a direct method to calculate kernels and prove that the complexity of computing revision is tractable.

In the area of Datalog+/-, Lukasiewicz *et al.* [4] give an inconsistency reasoning method for Datalog+/- ontologies, theirs work is close related to ours work since they use culprit to resolve the inconsistency of ontologies, and the culprit is equivalent with the kernel of this paper in the case of atom unwanted is $\bot$. However, there are some difference. Although the area of belief change is closely related to the management of inconsistent information, they are still quite different in both goals and constructions. Inconsistency can be handled by using kernels and clusters in [4], ours work can also deal with inconsistency, however revision operator given in ours work can do more except this, for example, it can choose a set of atoms as unwanted information, not only $\bot$, thus give more flexibility to resolve the inconsistency, and in this sense this work is more general than theirs work. Note also that in [4] the properties of the reasoning result are not clear even in the case of inconsistency handling because they did not study the properties of the operation from the viewpoint of belief revision.

There are still some works that extend Datalog+/- with the capability of dealing with uncertainty. In Lukasiewicz *et al.* [5], they developing a probabilistic extension of Datalog+/-. This extension uses Markov logic networks as the underlying probabilistic semantics and focus especially on scalable algorithms for answering threshold queries. Riguzzi *et al.* [8] apply the distribution semantics for probabilistic ontologies (named DISPONTE) to the Datalog+/- language. Lukasiewicz *et al.* [9] tackle the problem of defining a well-founded semantics for Datalog rules with existentially quantified variables in their heads and negations in their bodies, thus provide a kind of nonmonotonic reasoning capability to Datalog+/-. Our work also deal with the commonsense reasoning in the background of Datalog+/- language, however, we focus on the problem of belief revision, instead of adding quantitative or qualitative uncertainties to ontologies.

7 Summary and Outlook

In this paper, we address the problem of belief revision for Datalog+/- ontologies. In our approach, we introduce a kernel based belief revision operator, and study the properties using extended postulates, we then provide algorithms to revise Datalog+/- ontologies, and give the complexity results by showing that query answering for a revised linear Datalog+/- ontology is tractable.

In the future, we plan to study how to implement belief revision when some heuristic information, *i.e.*, different trust [12] or reputation [2] levels of both database and rule set due to the different source of information, can be added to Datalog+/- ontologies.

Acknowledgments. This work is partially supported by the National Natural Science Foundation of China Grant No.61003022 and Grant No.41174007, as well as the FP7 K-Drive project (No. 286348) and the EPSRC WhatIf project (No. EP/J014354/1).

References

1. Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis and Holger Wache. *Inconsistencies, Negations and Changes in Ontologies*. In Proc. of the 21st National Conference on Artificial Intelligence (AAAI-06), 1295-1300. 2006.
2. Andrew Koster and Jeff Z. Pan. *Ontology, Semantics and Reputation*. In Agreement Technologies, ISBN 978-94-007-5582-6, Springer. 2013.
3. Thomas Lukasiewicz, Andrea Cali and Georg Gottlob, *A General Datalog-Based Framework for Tractable Query Answering over Ontologies*. Journal of Web Semantics, 2012, Vol 14, Pages 57-83
4. Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Inconsistency Handling in Datalog+/- Ontologies*, in the Proceedings of the 20th European Conference on Artificial Intelligence (ECAI) 2012. Pages 558-563.
5. Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Query Answering under Probabilistic Uncertainty inDatalog+/- Ontologies*, Annals of Mathematics and Artificial Intelligence, 2013, Pages 195-197
6. Jeff Z. Pan, Edward Thomas, Yuan Ren and Stuart Taylor. *Exploiting Tractable Fuzzy and Crisp Reasoning in Ontology Applications*. In IEEE Computational Intelligence Magazine, 7(2):45-53. 2012.
7. Guilin Qi, Peter Haase, Zhisheng Huang, Qiu Ji, Jeff Z. Pan, Johanna Vlker. *A Kernel Revision Operator for Terminologies* . In Proc. of the 7th International Semantic Web Conference (ISWC2008). 2008.
8. Fabrizio Riguzzi, Elena Bellodi, and Evelina Lamma, *Probabilistic Datalog+/- under the distribution semantics*. In the Proceedings of the 25th International Workshop on Description Logics (DL), Aachen, Germany, 2012, pages 519-529.
9. Thomas Lukasiewicz, Maria Vanina Martinez and Gerardo I. Simari, *Well-Founded Semantics for Extended Datalog and Ontological Reasoning*. In the Proceedings of the 32nd ACM Symposium on Principles of Database System. ACM Press. 2013
10. Sven Hansson, *Semi-revision*, Journal of Applied Non-Classical Logics, Vol 7, Issue 1-2, 1997.
11. Marcio M. Ribeiro and Renata Wassermann. *Base Revision for Ontology Debugging*. Journal of Logic and Computation. Vol 19, Issue 5, 2009, pages 721-743.
12. Murat Sensoy, Achille Fokoue, Jeff Z. Pan, Timothy Norman, Yuqing Tang, Nir Oren and Katia Sycara. *Reasoning about Uncertain Information and Conflict Resolution through Trust Revision*. In Proc. of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2013). 2013.