A Distance-Based Operator to Revising Ontologies in DL $\mathcal{SHOQ}^{\star}$

Fangkai Yang[1], Guilin Qi[2], and Zhisheng Huang[3]

[1]Department of Computer Sciences, The University of Texas at Austin, USA
[2]AIFB-University of Karlsruhe, Karlsruhe, Germany
[3] Department of Computer Science, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Abstract. In this paper, we propose a distance-based operator to revise ontologies with acyclic generalized terminology as its TBOX in description logic $\mathcal{SHOQ}$. Our operator resolves incoherence between the original ontology and the newly received ontology. We first reformulate Dalal's operator to $\mathcal{SHOQ}$, and propose a query-equivalent syntactical formulation based on a notion called a *revision policy*. We then propose a tableau algorithm to generate such revision policies and prove the correctness of the algorithm. We show that the complexity of our algorithm stays at the same level as that of satisfiability check in $\mathcal{SHOQ}$.

1 Introduction

Ontology change is an important topic in the Semantic Web. When ontologies evolve, one of the central problems is to deal with logical contradictions [1,2]. When the newly received information is considered as more reliable or important than the original one, we will change the original ontology to resolve contradiction. In this case, this problem is similar to the problem of belief revision. Therefore, instead of proposing a solution from scratch, it is reasonable to reuse existing methods for belief revision to solve the problem of inconsistency handling in ontology change.

The most influential work on belief revision is done by Alchourrón, Gärdenfors and Makinson (AGM for short) who develop the so-called AGM theory of belief change [3]. However, it is not a trivial task to apply AGM theory to Description Logics (DLs) because some AGM assumptions fail for DLs. For example, the negation of a *terminology axiom* cannot be defined in most of DLs. In [4], AGM postulates for *contraction* are adapted to DLs, but they show that for some important DLs, such as $\mathcal{SHIQ}$ and $\mathcal{SHOIN}$, we cannot define a revision operator that satisfies all of their postulates. Furthermore, in [5], the authors propose a set of postulates for characterizing a revision operator for ontologies in DLs by introducing axiom negation in ontologies. They differentiate two kinds of logical contradictions in DLs during ontology change: inconsistency and incoherence. A DL-based ontology is inconsistent if it has no model and it is incoherent if there is a concept in the ontology which always denotes an empty set.

* Guilin Qi is partially supported by the EU in the IST project NeOn. Zhisheng Huang is partially supported by EU-funded Projects OpenKnowledge and LarKC. We thank the reviewers for very helpful comments to improve the quality of our work.

C. Sossai and G. Chemello (Eds.): ECSQARU 2009, LNAI 5590, pp. 434–445, 2009.

Indeed, incoherence is the logical error that occurs often in terminology part of a DL-based ontology, when new *terminology axioms* are added into the original ontology manually or automatically [6,7]. Although an incoherent ontology can have a model, querying over it may result in undesirable conclusions. The following scenario is adapted from [7], where new terminology axioms are added into an ontology PROTON which consists of a set of terminology axioms, through human annotation of the disjointness between two concepts. Suppose the following axioms are contained in PROTON: {HydrographicStructure $\sqsubseteq$ Facility $\sqcap$ Hydrographic, Reservoir $\sqsubseteq$ Lake $\sqcap$ HydrographicStructure, Lake $\sqsubseteq$ WaterRegion}. By learning that concept *WaterRegion* is disjoint with concept *Facility* and adding this disjointness axiom to the above ontology, we get an incoherence because concept *Reservoir* becomes unsatisfiable. Such a logical error, if not repaired, will result in trivial inference as any concept subsumed by concept *Reservoir* will become unsatisfiable. However, work on automatically resolving incoherence of ontologies in ontology change is rare.

In this paper, we propose a novel distance-based revision operator for ontologies in $\mathcal{SHOQ}$ [8] by adapting the well-known Dalal's revision operator [9], which is an intuitive, model-based revision operator satisfying all the AGM postulates. $\mathcal{SHOQ}$ is an expressive DL that underpins the Web ontology language OWL-DL [10], and our method is powerful in revising $\mathcal{SHOQ}$ ontology with *acyclic generalized terminology* as its TBOX [11]. As far as we know, it is the *first* revision operator that resolves incoherence by following the principle of minimal change and is not dependent on the syntactical forms of terminology axioms. Inspired by the work in [12], we propose a notion named *revision policy*, which first substitutes a concept name by a fresh concept name to resolve incoherence, and then asserts the cardinality difference between the new name and the original one to guarantee minimal change. Based on the revision policies, we can obtain an ontology which is query-equivalent to an ontology resulting from the revision operator. Finally, we propose an algorithm to generate such revision policies by extending the tableau algorithm for $\mathcal{SHOQ}$. We prove its correctness and show that the complexity our our algorithm stays at the same level as the complexity of satisfiability check in $\mathcal{SHOQ}$.

Proofs were omitted due to lack of space, but can be found in the technical report at http://www.cs.utexas.edu/∼fkyang/rev.pdf.

2 Description Logic $\mathcal{SHOQ}$

We assume that the reader is familiar with Description Logics (DLs) and refer to DL Handbook [11] (Chapter 2) for more details. In this section, we give a brief review of DL $\mathcal{SHOQ}$. Let $\mathbf{C}$, $\mathbf{R}_A$ and $\mathbf{I}$ be disjoint sets of *concept names*, abstract *role names*, and *individual names*. For R and S roles, a *role axiom* is either a role inclusion, which is of the form $R \sqsubseteq S$ for $R, S \in \mathbf{R}_A$ or a transitivity axiom, which is of the form $Trans(R)$ for $R \in \mathbf{R}_A$. A *RBOX* $\mathcal{R}$ is a set of role axioms. A role R is *simple* if, for $\trianglelefteq$ the transitive reflexive closure of $\sqsubseteq$ on $\mathcal{R}$ and for each role S, $S \trianglelefteq R$ implies $Trans(S) \notin \mathcal{R}$. The set of $\mathcal{SHOQ}$-concepts (or concepts) is the smallest set such that each concept name $A \in \mathbf{C}$ is a concept, for each individual name $o \in \mathbf{I}$, $\{o\}$ is a concept, and for C and D concepts, R an abstract role, S a simple role, complex concepts can be built using

conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), concept negation ($\neg C$), exists restriction ($\exists R.C$), universal restriction ($\forall R.C$), atleast restriction ($\geq n)S.C$, atmost restriction ($\leq n)S.C$. A *TBOX* is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are concepts. An *ABox* is a set of concept and role *assertions* $C(a)$, $R(a, b)$, and *(in)equality axioms* $a = b$ ($a \neq b$), where C is a concept, R is a role and a and b are individuals. An ontology is a triple $O = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBOX, $\mathcal{R}$ is an RBOX, and $\mathcal{A}$ is an ABOX. In our paper, O is considered as the set union of $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

The semantics of $\mathcal{SHOQ}$ ontology is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and the function $\cdot^{\mathcal{I}}$ which maps individuals, concepts and roles to elements of the domain, subsets of the domain and binary relations on the domain, respectively. For a complete definition of the semantics of $\mathcal{SHOQ}$ we refer the reader to [8]. An interpretation $\mathcal{I}$ is called a *model* of an ontology O, denoted as $\mathcal{I} \models O$, if it satisfies each axiom in the ontology. We use $\mathcal{M}(O)$ to denote the set of all models of ontology O. A concept C is unsatisfiable if for each model $\mathcal{I}$ of O, $C^{\mathcal{I}} = \emptyset$. An ontology O is incoherent if there exists an unsatisfiable concept in O and it is inconsistent if it has no model. Although incoherence is a notion different from classical view of inconsistency, it is firmly relevant with inconsistency [5]. That is, given an incoherent ontology O, and a set of concept assertions $A = \{C(i_C) | \text{for each concept name } C \ in O\}$, then $O^+ = O \cup A$, named as the enhanced ontology of O, is inconsistent. Furthermore, for nominal $\{o\}$, its unsatisfiability is defined to be a form of inconsistency, as we fail to find an interpretation for the individual o.

We are interested in the problem of incoherence handling in this paper. Since incoherence often occurs in terminologies, we assume all the ontologies consist of a TBOX and an RBOX, with empty ABOX, and all the logical contradictions take the form of incoherence. Furthermore, each TBOX has a restricted form, called *acyclic generalized terminology* (AGT) [11]. In a *generalized terminology*, each axiom is a GCI $C \sqsubseteq D$, where the left hand side is a concept name, which occurs at most once on the left hand side of axioms in the ontology. For each concept inclusion axiom $C \sqsubseteq D$, if a concept name C' occurs in D, we say C *uses* C', and the relation *uses* is transitive so that we can obtain a transitive closure. A generalized terminology is acyclic if any concept doesn't use itself. In the setting of ontology revision, given two ontologies O and O' which are represented as a set of DL axioms and assertions, we assume that the TBOX of $O \cup O'$ is an AGT. The concept names of O can be divided into two disjoint sets. $\mathcal{B}_{\mathcal{T}}$ are called *base symbols*, in which each concept, named a *primitive concept* is a concept name occurring only on the right hand side of the GCI, and $\mathcal{N}_{\mathcal{T}}$ are called *named symbols*, in which each concept, named a *defined concept*, is the symbol occurred on the left hand side of some axiom. Given the interpretations of symbols in $\mathcal{B}_{\mathcal{T}}$, we can build models of the terminology based on them.

Checking satisfiability of a $\mathcal{SHOQ}$ concept D is accomplished by a *tableau algorithm* [8], which tries to explicitly build up a model by the *completion forest* for the given concept and knowledge base by exhaustively applying a set of *tableau rules*. The algorithm initializes the completion forest F to contain $l + 1$ root nodes $x_0, x_{\{o_1\}}, \ldots, x_{\{o_l\}}$ with labels $\mathcal{L}(x_{o_i}) = \{\{o_i\}\}$, where o_i is nominal occurring in

D, and begin to expand the completion forest by applying two sets of rules: a set of *non-deterministic rules* $\mathbb{NR}$, i.e.$\sqcup$-rule, $choose$-rule, $\leq$ -rule, and the set of *deterministic rules* $\mathbb{DR}$, i.e, $\sqcap$-rule, $\exists$-rule, $\forall$-rule, $\forall_+$-rule, $\geq$-rule, **O**-rule, and terminates when the completion forest is *complete*: when either no more rules are applicable, where the concept is satisfiable, or all applications of the rules result in a *clash*, where inconsistency is met. Specifically, assuming there is no ABOX and no inconsistencies caused by nominals, a clash is of the following two forms: ($\mathtt{C}_1$): for some concept names $A \in N_C$, $\{A, \neg A\} \subseteq \mathcal{L}(x)$, and ($\mathtt{C}_2$):for some role names S, $(\leq n)S.C \in \mathcal{L}(x)$, and there are $n+1$ S-successors $y_0, \ldots, y_n$ of x with $C \in \mathcal{L}(y_i)$, for each $1 \leq i < j \leq n$ and $y_i \neq y_j$.

3 A Semantic Revision Operator in $\mathcal{SHOQ}$

3.1 Definition

Our revision operator is based on the well-known Dalal's operator [9]. The idea of this revision operator is that the models of the revised knowledge base of the operator should be the models of the newly received knowledge base which have minimal distance with the original one. However, adapting such idea to DLs is not trivial, because DLs have first-order features. Following the idea of Dalal's operator, we first define the distance between two interpretations and use it to define the distance between an interpretation and an ontology.

Definition 1. *(Distance between interpretations) Let $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \Delta, \cdot^{\mathcal{I}'} \rangle$ be two interpretations over the same domain. Let $d(M^{\mathcal{I}}, M^{\mathcal{I}'}) = |M^{\mathcal{I}} \ominus M^{\mathcal{I}'}|$ where M is a concept name or a role name. The distance between $\mathcal{I}$ and $\mathcal{I}'$, denoted $d(\mathcal{I}, \mathcal{I}')$, is defined as follows:*

$$d(\mathcal{I}, \mathcal{I}') = \sum_{A \in L_C} |A^{\mathcal{I}} \ominus A^{\mathcal{I}'}| + \sum_{R \in L_R} |R^{\mathcal{I}} \ominus R^{\mathcal{I}'}|$$

where $S \ominus S'$ denotes the symmetric difference between sets S and S', i.e., $S \ominus S' = (S \cup S') \setminus (S \cap S')$, L_C and L_R are respectively the sets of all concept names and role names which are used to construct the DL ontology.

Definition 2. *Let $O = \langle \mathcal{T}, \mathcal{A} \rangle$ and $O' = \langle \mathcal{T}', \mathcal{A}' \rangle$ be ontologies with empty ABOXes. Let $\mathcal{I}$ be a model of O'. The distance between $\mathcal{I}$ and O, denoted $d(\mathcal{I}, O)$, is defined as follows: $d(\mathcal{I}, O) = min_{\mathcal{I}' \models O} d(\mathcal{I}, \mathcal{I}')$, where $\mathcal{I}$ and $\mathcal{I}'$ are over the same domain.*

Based on the above definition, we can define a total pre-order on the models of O' as follows: $\mathcal{I} \preceq_O \mathcal{I}'$ iff $d(\mathcal{I}, O) \leq d(\mathcal{I}', O)$. We can also define the distance between these two ontologies as $d(O, O') = min_{\mathcal{I} \models O'} d(\mathcal{I}, O)$.

Incoherence doesn't lead to the classical sense of contradiction: we may have $d(O, O') = 0$ if $O \cup O'$ is incoherent but consistent. Therefore, unlike Dalal's operator, we cannot define the models of the revised ontology as the models of O' which are minimal w.r.t. the ordering $\preceq_O$. Instead, we append a fresh individual to each concept in the ontology to render inconsistency so that we can apply the idea of Dalal's operator to define a revision operator for resolving incoherence. Based on the notion of

enhanced ontologies in [5], we define the *enhancement* of O relative to O' as $O^+_{O'} = O \cup \{C(i_C) | \text{for all the primitive concepts } C \text{ of } O \cup O', i_C \text{ is a fresh name for } C\}$.

Definition 3. *Let O and O' be two ontologies, and $O^+_{O'}$ and $(O')^+_O$ be their enhanced forms. The result of revision of O by O', denoted $O \circ_D O'$, is defined in a model-theoretical way as follows:*

$$\mathcal{M}(O \circ_D O') = Min(\mathcal{M}((O')^+_O), \preceq_{O^+_{O'}}).$$

That is, the models of the result of revision of O with O' are models of enhanced ontology of O' that are minimal w.r.t. the total pre-order $\preceq_{O^+_{O'}}$.

Example 1. For $O = \{C \sqsubseteq \forall R.D\}$ and $O' = \{C \sqsubseteq \exists R.\neg D\}$, $O^+_{O'} = O \cup \{C(i_D), D(i_D)\}$ $(O')^+_O = O' \cup \{C(i_D), D(i_D)\}$. Therefore, given $\Delta = \{a, b, c, d, o_C, o_D\}$, let $C^{\mathcal{I}} = \{a, b, o_C\}$, $D^{\mathcal{I}} = \{c, d, o_D\}$, and $R^{\mathcal{I}} = \{(a, c), (b, d), (o_C, o_D)\}$, $(i_C)^{\mathcal{I}} = o_C$ and $(i_D)^{\mathcal{I}} = o_D$, we have $\mathcal{I} \models O^+_{O'}$. Let $C^{\mathcal{I}'} = \{a, b, o_C\}$, $D^{\mathcal{I}'} = \{c, d\}$, and $R^{\mathcal{I}'} = \{(a, c), (b, d), (o_C, o_D)\}$, $(i_C)^{\mathcal{I}'} = o_C$ and $(i_D)^{\mathcal{I}'} = o_D$ $\mathcal{I}' \models (O')^+_O$. So $d(\mathcal{I}, \mathcal{I}') = 1$. Therefore, $\mathcal{I}' \in Min(\mathcal{M}((O')^+_O), \preceq_{O^+_{O'}})$ and thus $\mathcal{I}' \in \mathcal{M}(O \circ_D O')$

Our revision operator leads to a coherent ontology while preserving the consistency of the resulting ontology.

Theorem 1. *Let O and O' be two ontologies with empty ABOXes and O' be consistent and coherent, then $O \circ_D O'$ is coherent and consistent.*

3.2 Syntactic Formulation of Our Revision Operator

The syntactical formulation is inspired by the work in [12], in which a propositional knowledge base K revised relative to a propositional formula ϕ using Dalal's operator is query-equivalently formulated as a cardinality-circumscription theory, where each atomic proposition of K, say p, is substituted by a fresh name, say p', and a fresh proposition w defined as $p = p'$ is cardinality circumscribed. When dealing with first order semantics, we formalize it by a notion of *revision policy*.

Definition 4. *(Substitution) A substitution ϕ on ontology O is defined as $[C/C']$ where C is the concept occurring in O and C' is a fresh concept.*

Given $\phi = [C/C']$, we use $O\phi$ to denote an ontology obtained by substituting each occurrence of C in O by C'. For two substitution $\phi_1 = [C/C']$ and $\phi_2 = [D/D']$ where C, D are different concept names in O and C', D' are fresh concept names, the composition of $\phi_1 \circ \phi_2$ is defined as $[C, D/C', D']$ and we have $O(\phi_1 \circ \phi_2) \doteq (O\phi_1)\phi_2$.

We now define the revision policy which is critical to the computation of our revision operator.

Definition 5. *(Revision Policy) Given ontology O, a revision policy $\mathbb{P}$ is a pair $< \phi, n_\phi >$ where $\phi = [C/C']$ is a substitution on a primitive concept C in O, and n_ϕ is an integer, called the degree of $\mathbb{P}$. The ontology $O\mathbb{P}$ obtained by applying $\mathbb{P}$ to O is defined in a model-theoretical way as: $\mathcal{I}' \models O\mathbb{P}$ iff $\mathcal{I}' \models O\phi$ and there exists a model $\mathcal{I}$ of O such that (1) $C^{\mathcal{I}'} \ominus C^{\mathcal{I}} = n_\phi$, (2) $C'^{\mathcal{I}'} = C^{\mathcal{I}}$, and (3) $D^{\mathcal{I}'} = D^{\mathcal{I}}$), for any other concept name D in O which is different from C.*

Imposing an revision policy $\mathbb{P} =< [C/C'], n >$ to O will result in another ontology $O\mathbb{P}$ with fresh concept C' such that (1) interpretation of C' in $O\mathbb{P}$ is the same as that of C in O, and the interpretation of C in $O\mathbb{P}$ has a distance of n from that in O.It is intuitive to resolve incoherence by changing the interpretation of primitive concepts because unsatisfiability of defined concepts are usually caused by primitive concepts. The following theorem states the validity of the definition.

Theorem 2. *Given an ontology O and a revision policy $\mathbb{P} =< C/C', n_\phi >$, for every model $\mathcal{I}$ of O, there exists a model $\mathcal{I}'$ of $O\mathbb{P}$ such that $C^{\mathcal{I}'} \ominus C^{\mathcal{I}} = n_\phi$, $D^{\mathcal{I}'} = D^{\mathcal{I}}$, for other concept name D in $O[C/C']$, and $(C')^{\mathcal{I}'} = C^{\mathcal{I}}$.*

Given two revision policies $\mathbb{P} =< \phi, n_\phi >$ and $\mathbb{Q} =< \xi, n_\xi >$, where $\phi = [C/C']$, $\xi = [D/D']$ and C, D are not synonyms, the composition of $\mathbb{P}$ and $\mathbb{Q}$ is defined as $\mathbb{P} \circ \mathbb{Q} \doteq < \phi \circ \xi, \{n_\phi, n_\xi\} >$.

Definition 6. *Given an ontology O, the ontology $O(\mathbb{P} \circ \mathbb{Q})$ obtained by applying $\mathbb{P} \circ \mathbb{Q}$ to O is defined in a model-theoretical way as: $\mathcal{I}' \models O\mathbb{P} \circ \mathbb{Q}$ iff $\mathcal{I}' \models O\phi \circ \xi$ and there exists a model $\mathcal{I}$ of O such that (1) $C^{\mathcal{I}'} \ominus C^{\mathcal{I}} = n_\phi$ and $D^{\mathcal{I}'} \ominus D^{\mathcal{I}} = n_\xi$, (2) $C'^{\mathcal{I}'} = C^{\mathcal{I}}$ and $D'^{\mathcal{I}'} = D^{\mathcal{I}}$ for new concept names C', D', (3) $E^{\mathcal{I}} = E^{\mathcal{I}'}$, for any concept name E in O different from C and D. For two revision policies with same substitution, i.e., $\mathbb{P} =< \phi, n_\phi >$ and $\mathbb{Q} =< \phi, n_\xi >$, $\mathbb{P} \circ \mathbb{Q} \doteq < \phi, max\{n_\phi, n_\xi\} >$.*

It is easy to check that for any revision policies $\mathbb{P}$ and $\mathbb{Q}$ and ontology O, we have $\mathcal{M}(O(\mathbb{P} \circ \mathbb{Q})) = \mathcal{M}(O(\mathbb{P})\mathbb{Q}) = \mathcal{M}(O(\mathbb{Q} \circ \mathbb{P}))$.

We now consider the syntactical counterpart of revision policy. For ontology O on language $\mathcal{L}$, given a revision policy $\mathbb{P} =< [C/C'], n_C >$, we can obtain an axiom set $\mathcal{A}_{\mathbb{P}}$ consisting of the following axioms $\{o_1, \ldots, o_{|n_C|}\} \equiv ((C \sqcap \neg C') \sqcup (\neg C \sqcap C'))$, where $o_1, \ldots, o_{|n_C|}$ are fresh nominals not occurring in $\mathcal{L}$, and furthermore, we assume unique name assumption (UNA) on them.

The following theorem states that a revision policy $\mathbb{P}$ can be syntactically characterized by substitution and axiom set $\mathcal{A}_{\mathbb{P}}$.

Theorem 3. *Given ontology O, for a revision policy $\mathbb{P} =< \phi, n_\phi >$ where $\phi = [C/C']$, we have $\mathcal{M}(O\mathbb{P}) = \mathcal{M}(O\phi \cup \mathcal{A}_{\mathbb{P}})$.*

By Theorem 3, we have the following corollary.

Corollary 1. *Given ontology O, for two revision policy $\mathbb{P} =< \phi, n_\phi >$ and $\mathbb{Q} =< \xi, n_\xi >$, where ϕ and ξ only substitute the symbols occurring in $\mathcal{L}$. We have $O(\mathbb{P} \circ \mathbb{Q}) = O(\phi \circ \xi) \cup \mathcal{A}_{\mathbb{P}} \cup \mathcal{A}_{\mathbb{Q}}$.*

Now we characterize the query-equivalent syntactical counterpart of the operator by revision policies, beginning with the following definition and lemma.

Definition 7. *Given two ontologies O, O' and models $\mathcal{I}$ and $\mathcal{I}'$ of O and O' respectively such that $d(\mathcal{I}, \mathcal{I}') = d(O^+_{O'}, O'^+_O)$, for primitive concept C such that $C^{\mathcal{I}} \neq C^{\mathcal{I}'}$. We say $\mathbb{P}_i$ is* generated *from $\mathcal{I}$ and $\mathcal{I}'$ if $\mathbb{P}_i =< [C/C'], |C^{\mathcal{I}} \ominus C^{\mathcal{I}'}| >$.*

Lemma 1. *Let $O \circ_D O' \models C \sqsubseteq D$ for GCI $C \sqsubseteq D$. For all revision policies $\mathbb{P}_1, \ldots, \mathbb{P}_n$ generated from model $\mathcal{I}$ of O, and $\mathcal{I}'$ of O' such that $d(\mathcal{I}, \mathcal{I}') = d(O^+_{O'}, O'^+_O)$, they satisfies (1) $\Sigma_{i=1}^{n} deg(\mathbb{P}_i)$ is minimal; (2) $O\mathbb{P}_1 \ldots \mathbb{P}_n \cup O'$ is coherent and consistent, and (3) $O\mathbb{P}_1 \ldots \mathbb{P}_n \cup O' \models C \sqsubseteq D$.*

Theorem 4. *Given ontology O and O' and a query of the form $C \sqsubseteq D$, where C and D are concepts, then $O \circ_D O' \models C \sqsubseteq D$ if and only if for any sequence of revision policies $\mathbb{P}_1, \ldots, \mathbb{P}_n$ on $\mathcal{L}$ such that $\sum_{1 \leq k \leq n} n_{\mathbb{P}_k}$ is minimal and $(O\mathbb{P}_1 \ldots \mathbb{P}_n) \cup O'$ is coherent and consistent, $(O\mathbb{P}_1 \ldots \mathbb{P}_n) \cup O' \models C \sqsubseteq D$.*

4 A Tableau Algorithm for Policy-Based Revision

When an ontology is incoherent, its completion forest can only have clashes $(\mathtt{C}_1)$, $(\mathtt{C}_2)$ given in Section 2. Our method focuses on resolving the clashes by extending the tableau algorithm of $\mathcal{SHOQ}$ with several *repairing rules* so that no clash can be met. The strategy for such repair is to generate revision policies with minimal degree.

Specifically, given ontologies O that is to be revised by O', suppose that concept D is unsatisfiable in $O \cup O'$, the completion forest of D relative to $O \cup O'$ will contain clashes. Informally, we repair the above two kinds of clashes as follows.

- **Concept Clash Repair.** For clash $(\mathtt{C}_1)$, we rename A by A', and specify the difference between the interpretations of A and A' as 1. Therefore, we generate a revision policy $\mathbb{P} =< [A/A'], 1 >$ (see R_1 in Fig.1).
- **Role Clash Repair.** For clash $(\mathtt{C}_2)$, we rename C by C', and specify the difference between the interpretations of C and C' as 1. Therefore, we generate a revision policy $\mathbb{P} =< [C/C'], 1 >$. If there are more than $n+1$ different S-successors of x, then we increase the degree of $\mathbb{P}$ (see R_2 in Fig.1).

We extend the tableau algorithm to repair the clashes in the completion forests. For concept C in an ontology, we arrange all of its completion forests into a *hyper-tree*, in which each node is a *weak complete* completion forest, and each leave is a complete completion forest in the sense of [8]. If C is unsatisfiable, then each leaf contains clash(es), from which the revision policies will be generated. First we define the notion of *weak-completeness* for the completion forest.

Definition 8. *Given a $\mathcal{SHOQ}$ concept D in Negation Normal Form (NNF), a completion forest is weak-complete iff all rules in $\mathbb{DR}$ have been applied till no more of these rules are applicable and none of the nondeterministic rules in $\mathbb{NR}$ has ever been applied.*

From a weak completion tree, we then use rules in $\mathbb{NR}$ to generate its *successors* by first duplicating all the items from a node into a new one and then using nondeterministic rules in $\mathbb{NR}$ to make it weak-complete.

Definition 9. *Given two completion forests F_1 and F_2 which are weak- complete, F_2 is the successor of F_1, denoted as $succ(F_1, F_2)$, if (1) F_1 is weak-complete; (2) F_2 is generated from F_1 by first copying all the nodes of F_1, the structure between the nodes,*

and the labels of the nodes, and then exhaustively applying rules in $\mathbb{NR}$, and (3) F_2 is weak completed by rules in $\mathbb{DR}$. A node F_1 is successor complete *if all of its successors have been generated.*

Finally, we define the completion hyper-tree, which organizes a set of weak completion forests by the above successor relationship.

Definition 10. *Given a $\mathcal{SHOQ}$ concept D, the completion hyper-tree $\mathbb{T}$ of D is inductively defined as follows:*

1. *The root of $\mathbb{T}$, denoted as $root(\mathbb{T})$ is initialized in the same way as tableau algorithm of $\mathcal{SHOQ}$ does, and then completed to be weak-complete.*
2. *The subtrees of $root(\mathbb{T})$ are denoted as $\mathbb{T}_1, \ldots, \mathbb{T}_n$, and $succ(root(\mathbb{T}), root(\mathbb{T}_i))$ holds for $1 \leq i \leq n$, and $\mathbb{T}_i$ are all completion hyper-trees.*

A completion hyper-tree is *complete* if (1) all of its non-leaf nodes are weak complete and successor complete, and (2) all of the leaf nodes are complete in the sense of tableau algorithm of $\mathcal{SHOQ}$. When there are clashes in the set F of leaves of the hyper-tree, for each node x_i of F, R_1 and R_2 in Fig.1 will compute revision policies resolving each clash. However, this is not enough because the concept influenced by the revision policy may occur in O' rather than O, which has no effect to O, as the following example illustrates.

Example 2. Given $O = \{A = C_1 \sqcap C_2\}$, and $O' = \{C_1 \sqsubseteq D, C_2 \sqsubseteq \neg D\}$, we have a clash in the completion forest of A relative to $O \cup O'$ as $\{D, \neg D\}$, and by revision policy $\mathbb{P} =< [D/D'], 1 >$ we can repair this clash. However, we find that $\mathbb{P}$ repairs concepts in O' rather than O. To deal with this case, we need to be aware that the clash is caused by C_1 and C_2, which occur in O, and that $C_2 \sqsubseteq \neg C_1$ (or $C_2 \sqsubseteq \neg C_1$). Instead, we can use $< [C_1/C_1'], 1 >$ (or $< [C_2/C_2'], 1 >$) to repair the defined concept which makes another concept involved in a clash.

The above example shows that if those concepts in the clash of the completion forrest happen to be those in O' but not in O, we need to revise the concepts in O that are *dependent* on concepts in O' that are involved in the clash. Based on this observation, we define the notion of *concept dependency* in AGT.

Definition 11. *Given ontology O, we use $\boldsymbol{C}$ to denote a set of concept names appearing in O. Let $\boldsymbol{C}^+ = \boldsymbol{C} \cup \{\neg C | C \in \boldsymbol{C}\} \cup \{\{o\} | \text{for individual } o \; in O\}$. A dependency relation $\mathcal{D}$ is defined as a binary relation on $\boldsymbol{C}^+ \times \boldsymbol{C}^+$ such that given two concepts C and C' in $\boldsymbol{C}^+$, we say that C is dependent on C' if there exists a specification $C \sqsubseteq D$ such that C' appearing in D. We use $\mathcal{D}$ to denote the dependence relation.*

For each leaf of a completion hyper-tree which is complete, we can create a dependency graph by Algorithm 1 with $\bot$ on the top and $\top$ at the bottom. Based on the dependency graph, we propose two *tracing rules* (see Fig. 1). As we assume that $O \cup O'$ is an AGT and both O and O' are coherent, tracing rules can always find a concept occurring in O, otherwise O' itself is incoherent. Furthermore, the first concept it finds must be a primitive concept in O, otherwise $O \cup O'$ is not an AGT.

Algorithm 1. Generating Dependency Graph

Procedure CreateDependencyGraph($\mathbb{T}$)
$\mathbf{C}^+ := \{C | C \in \mathcal{L}(x)\}$, where x is the node of $\mathbb{T}$.
$\mathcal{D}_\mathbb{T} := \emptyset$
for all all $C \in \mathbf{C}^+$ such that there exists no $C' \in \mathbf{C}^+$ which is dependent on C **do**
    $\mathcal{D}_\mathcal{T} := \{(\bot, C)\} \cup \mathcal{D}_\mathbb{T}$
    CreateGraph($C, \mathbf{C}^+, \mathcal{D}_\mathbb{T}$)
end for
Procedure CreateGraph($C, \mathbf{C}^+, \mathcal{D}_\mathbb{T}$)
for all concept $C_i \in \mathbf{C}^+$ **do**
    if C is dependent on C_i and $(C, C_i) \notin \mathcal{D}_\mathbb{T}$ **then**
        $\mathcal{D}_\mathbb{T} := \mathcal{D}_\mathbb{T} \cup (C, C_i)$
        CreateGraph($C_i, \mathbf{C}^+, \mathcal{D}_\mathbb{T}$)
    else
        $\mathcal{D}_\mathbb{T} := \{(C, \top)\} \cup \mathcal{D}_\mathbb{T}$
    end if
end for

Based on the above algorithm, we can see that the dependency relation $\mathcal{D}$ for Example 2 includes (A, C_1), (A, C_2), (C_1, D) and $(C_2, \neg D)$. The following theorem states that Algorithm 1 terminates in polynomial time.

Theorem 5. *Suppose $\mathbb{T}$ is a leaf of a complete completion hyper-tree, the algorithm $CreateDependencyGraph(\mathbb{T})$ terminates within polynomial time relative to $|\boldsymbol{C}^+|$, returning a directed acyclic dependency graph $\mathcal{D}$.*

Extending the tableau algorithm with R_1-rule, R_2-rule, tracing rule-1 and tracing rule-2 in Figure 1, we can obtain a set of revision policies $\mathbb{P}_1, \ldots, \mathbb{P}_n$ addressing each kind of clash. To repair all the clashes occurring in a completion forest, we need the composite revision policy:

$$\mathbb{P} = \mathbb{P}_1 \circ \ldots \circ \mathbb{P}_n \doteq \prod_{1 \le i \le n} \mathbb{P}_i \tag{1}$$

where each revision policy contains different substitution. We will later prove that the above rules only make minimal change to the difference between the introduced concept and the original concept. For a completion forest F, and for all the nodes x_i of F with revision policy sets $\mathcal{S}^{x_i}$, we have

$$\mathbb{P}_F = \prod_{x_i \in F} \Big(\prod_{\mathbb{P}^{x_i} \in \mathcal{S}^{x_i}} \mathbb{P}^{x_i} \Big), \mathbb{D}_F = \sum_{x_i \in F} \Big(\sum_{\mathbb{P}^{x_i} \in \mathcal{S}^{x_i}} deg(\mathbb{P}^{x_i}) \Big) \tag{2}$$

For any concept D, it usually has more than one completion forest, due to the existence of nondeterministic rules in $\mathbb{DR}$. Given all completion forests with their revision policies, we will choose those whose degrees are the minimal:

$$\mathbb{P} = min_{\mathbb{D}_{F_i}} \{\mathbb{P}_{F_i} | 0 \le i \le n\} \tag{3}$$

In Fig.1, composite rule composes all the revision policies for each node to obtain the revision policy of F. Finally, synthesis rule chooses the revision policy with smallest degree at each branch of the hyper-tree. See the following example.

R_1-rule	if $C, \neg C \in \mathcal{L}(x)$, then $\mathbb{P} =< [C/C'], 1 >$
R_2-rule	if $(\leq n)S.C \in \mathcal{L}(x)$, and there are $n+1$ S-successors $y_0, \ldots, y_n$ of x, $C \in \mathcal{L}(y_i)$, for each $1 \leq i < j \leq n$ and $y_i \neq y_j$, $Rel^{\neq} \Leftarrow Rel^{\neq} \backslash \{y_i \neq y_j\}$, and if there is no revision policy for C, then $\mathbb{P} = [< C/C' >, 1]$; else $\mathbb{P} := increment(\mathbb{P})$. $Rel^{\neq}$ is the binary relation recording all inequalities between constants inherited from Tableau algorithm.
tracing-rule-1	For a revision policy $\mathbb{P} =< C/C', n >$ generated by R_1-rule, if C occurs in O', trace in the dependency graph of the completion tree to the nearest concept D such that D occurs in O and D is dependent on C or $\neg C$, and change $\mathbb{P}$ to be $< D/D', 1 >$.
tracing-rule-2	For a revision policy $\mathbb{P} =< C/C', n >$ generated by R_2-rule, if C occurs in O', trace in the dependency graph of the completion tree to the nearest concept D such that D occurs in O and D is dependent on C, and change $\mathbb{P}$ to be $< D/D', 1 >$.
composition-rule	For a node F, for all the node x_i of F with the revision policy sets $\mathcal{S}^{x_i}$ of, $\mathbb{P}_F = \prod_{x_i \in F}(\prod_{\mathbb{P}^{x_i} \in \mathcal{S}^{x_i}} \mathbb{P}^{x_i})$, and $\mathbb{D}_F = \sum_{x_i \in F}(\sum_{\mathbb{P}^{x_i} \in \mathcal{S}^{x_i}} deg(\mathbb{P}^{x_i}))$
synthesis-rule	For a node F of $\mathbb{T}_D$ and all its successors $F_1, \ldots, F_n \in \mathbb{T}_D$, $\mathbb{P}_F = min_{\mathbb{D}_{F_i}}\{\mathbb{P}_{F_i} \mid 0 \leq i \leq n\}$
Note: For $\mathbb{P} =< [C/C'], n >$, $increment(\mathbb{P}) =< [C/C', n+1] >$	

Fig. 1. The tableaux of revision policy generation

Example 3. We consider ontologies O and O' adapted from PROTON:

O	Reservoir$\sqsubseteq$ Lake$\sqcap$ HydrographicStructure, Lake$\sqsubseteq$ NaturalWaterRegion HydrographicStrure$\sqsubseteq (\geq 3)$ Owns.Harbor $\sqcap$ Hydrographic
O'	NaturalWaterRegion$\sqsubseteq (\leq 1)$ Own.Harbor

By applying R_2, we can obtain a revision policy $\mathbb{P} =< [Harbor/Harbor'], 1 >$. The revised ontology will be $O\mathbb{P} \cup O' = O[Harbor/Harbor'] \cup O' \cup \{\{o_1, o_2\} \equiv ((Harbor \sqcap \neg Harbor') \sqcup (\neg Harbor \sqcap Harbor'))\}$. In this case, *Reservoir* is satisfiable. As $\mathbb{P}$ is the only revision policy generated, for query-answering on $O\mathbb{P} \cup O'$, we have Reservoir$\sqsubseteq (\leq 1)$Owns.Harbor and HydroGraphicStructure$\sqsubseteq$Facility now becomes unknown.

However, HydroGraphicStructure$\sqsubseteq$HydroGraphic still holds, illustrating the syntax irrelevance of our method. Furthermore, we can also obtain Reservoir$\sqsubseteq (= 1)$ Owns.Harbor. Intuitively, the introduced name Harbor$'$ can be regarded as an *abnormal Harbor*, which is different from Harbor with one individuals. Such advantage can also benefit to build consistent ABOX afterwards: for each individual of Reservoir, it can only have one Harbor. If it is connected with more harbors, they are inferred to be synonyms.

We now discuss properties of the extended tableau for revising a $\mathcal{SHOQ}$ ontology.

Theorem 6. *A $\mathcal{SHOQ}$-concept C in NNF is satisfiable wrt a RBOX $\mathcal{R}$ if and only if the expansion rules can yield a complete completion hyper-tree, and at least one of its leaves does not contain revision policies.*

Theorem 7. *Given a $\mathcal{SHOQ}$ ontology O and a concept C, the extended tableau algorithm, when applied to C, will terminate in finite steps.*

The following theorem shows the correctness of our algorithm.

Theorem 8. *Given two $\mathcal{SHOQ}$ ontologies O and O', and an unsatisfiable concept C in O in $O \cup O'$. Let the complete completion hyper-tree of $O \cup O'$ be $\mathbb{T}_D$. Then the revision policy of $\mathbb{T}_D$ is $\mathbb{P}_D$ if and only if $deg(\mathbb{P}_D)$ is minimal relative to all revision policies $\mathbb{P}$ such that D is satisfiable in $O_1\mathbb{P} \cup O_2$.*

The algorithm is applied multiple times to repair all unsatisfied concepts. As the revision policies can be generated based on the framework of the tableau algorithm of $\mathcal{SHOQ}$, we have:

Theorem 9. *Given a query Q, checking $O \circ_D O' \models Q$ is EXPTime-complete.*

5 Related Work

The work in [4,5] focus on the postulates of rational revision operators in DLs and no concrete revision operator is given. In [13], two revision operators are given to revise ontologies in DL $\mathcal{ALCO}$ but they only consider the inconsistencies due to objects being explicitly introduced in the ABOX and their operators are syntax-dependent. Unlike the AGM-oriented approaches, the revision operators presented in [1,14,15] delete some elements from the original ontology to accommodate the new ontology and so they are all syntax-dependent. In [5], the authors argue that incoherence is also important during revision. However, they do not give a revision operator to deal with this problem. The work on debugging and repairing (see, for example, [6,16]) may be applied to give a revision operator that can resolve incoherence. However, these approaches are also syntax-dependent. In contrast, our revision operator is syntax-independent. Our work is also related to the work on updating ABOX in DLs [17] where an ordering between interpretations w.r.t. an interpretation is given and this ordering is used to define an update operator. In contrast, we define an ordering on the interpretations based on a distance function which is not dependent on a specific interpretation and use this ordering to define our operator.

6 Conclusion and Future Work

In this paper, we proposed a novel revision operator for $\mathcal{SHOQ}$ ontologies by adapting Dalal's revision operator. Since a straightforward adaption does not work, we used the notion of enhancement of an ontology to define our revision operator. We then proposed a notion named revision policy to obtain an ontology which is query-equivalent to an ontology resulting the revision operator. We extended the tableau algorithm for DL $\mathcal{SHOQ}$ by proposing some novel rules to generate revision policies to resolve clashes in the original tableau. We showed that our algorithm is correct and that the complexity our our algorithm stays at the same level as the complexity of satisfiability check in $\mathcal{SHOQ}$. Our framework can be easily adapted to OWL-DL or even more expressive language such as $\mathcal{SHOIQ}$.

As a future work, we will consider other revision operators such as Satoh's operator [12]. However, this problem is very challenging because Satoh's operator is not based on cardinality-circumscription. We are also considering applying this work into real Semantic Web system.

References

1. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 182–197. Springer, Heidelberg (2005)
2. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 353–367. Springer, Heidelberg (2005)
3. Gärdenfors, P.: Knowledge in Flux-Modeling the Dynamic of Epistemic States. MIT Press, Cambridge (1988)
4. Flouris, G., Plexousakis, D., Antoniou, G.: On applying the agm theory to dls and owl. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 216–231. Springer, Heidelberg (2005)
5. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proc. of AAAI 2006, pp. 1295–1300 (2006)
6. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. J. Autom. Reasoning 39(3), 317–349 (2007)
7. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)
8. Horrocks, I., Sattler, U.: Ontology reasoning in the shoq(d) description logic. In: Proc. of IJCAI 2001, pp. 199–204 (2001)
9. Dalal, M.: Investigations into a theory of knowledge base revision: Preliminary report. In: Proc. of AAAI 1988, pp. 475–479 (1988)
10. Patel-Schneider, P., Hayes, P., Horrocks, I.: Owl web ontology language semantics and abstract syntax. In: Technical report, W3C. W3C Recommendation (2004)
11. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, implementation and application. Cambridge University Press, Cambridge (2007)
12. Liberatore, P., Schaerf, M.: Reducing belief revision to circumscription (and vice versa). Artif. Intell. 93, 261–296 (1997)
13. Qi, G., Liu, W., Bell, D.A.: Knowledge base revision in description logics. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS, vol. 4160, pp. 386–398. Springer, Heidelberg (2006)
14. Halaschek-Wiener, C., Katz, Y., Parsia, B.: Belief base revision for expressive description logics. In: Proc. of OWL-ED 2006 (2006)
15. Ribeiro, M.M., Wassermann, R.: Base revision in description logics - preliminary results. In: Proc. of IWOD 2007 (2007)
16. Qi, G., Haase, P., Huang, Z., Ji, Q., Pan, J.Z., Völker, J.: A kernel revision operator for terminologies - algorithms and evaluation. In: Proc. of ISWC 2008, pp. 419–434 (2008)
17. de Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: Proc. of AAAI 2006 (2006)