Computing Minimum Cost Diagnoses to Repair Populated DL-based Ontologies

Jianfeng Du
State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
Graduate University of the
Chinese Academy of Sciences
Beijing 100080, China
jfdu@ios.ac.cn

Yi-Dong Shen
State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
Beijing 100080, China
ydshen@ios.ac.cn

ABSTRACT

Ontology population is prone to cause inconsistency because the populating process is imprecise or the populated data may conflict with the original data. By assuming that the intensional part of the populated DL-based ontology is fixed and each removable ABox assertion is given a removal cost, we repair the ontology by deleting a subset of removable ABox assertions in which the sum of removal costs is minimum. We call such subset a *minimum cost diagnosis*. We show that, unless P=NP, the problem of finding a minimum cost diagnosis for a DL-Lite ontology is insolvable in PTIME w.r.t. data complexity. In spite of that, we present a feasible computational method for more general (i.e. $\mathcal{SHIQ}$) ontologies. It transforms a $\mathcal{SHIQ}$ ontology to a set of disjoint propositional programs, thus reducing the original problem into a set of independent subproblems. Each such subproblem computes an optimal model and is solvable in logarithmic calls to a SAT solver. Experimental results show that the method can handle moderately complex ontologies with over thousands of ABox assertions, where all ABox assertions can be assumed removable.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Algorithms

Keywords

Ontologies, Description Logics, Disjunctive Datalog, Diagnosis

1. INTRODUCTION

Nowadays OWL [22] has been established as a core standard in the Semantic Web. It comes in three layers in ascending expressivity, i.e., OWL Lite, OWL DL and OWL Full, where the former two coincide semantically with certain description logics (DLs) [1]. A DL-based ontology consists of an intensional part and an extensional part. The *intensional part* consists of a TBox and an RBox, and contains knowledge about concepts and relations (called *roles*) between the elements of the domain. The *extensional part* consists of an ABox, and contains knowledge about individuals and how they relate to the concepts and roles from the intensional part. In this paper, the knowledge in intensional parts is called *axioms*, whilst the knowledge in extensional parts is called *ABox assertions* or simply *assertions*.

A crucial question in the vision of Semantic Web is how to support and ease the process of creating and maintaining DL-based ontologies. An important task within this process is *ontology population*, which adds instances of concepts and relations to the ontology. In recent years, there has been a great surge of interest in methods for populating ontologies from textual resources. To name a few, Text2Onto [3] and KITE [30] are frameworks that integrate algorithms for populating ontologies from textual data. The algorithms include information extraction algorithms that assign annotations carrying some semantics to regions of the data, and co-reference algorithms that identify annotated individuals in multiple places. As for populating DL-based ontologies, the information extraction process behaves as adding concept/role assertions, whilst the co-reference process behaves as adding equality/inequality assertions. The populated ontology, however, may become inconsistent because the information extraction/co-reference process is imprecise or the populated data possibly conflict with the original data. In order to repair the populated ontology, we propose to delete a subset of assertions in which the sum of removal costs is minimum, based on the following considerations. First, the intensional part should not be changed, because in general it is well prepared and *coherent* (i.e., having no unsatisfiable concepts) before the population. Second, for changing the extensional part, only the deletion of assertions is considered because there is generally no criterion for revising assertions. Third, for deleting assertions, some minimal criteria on removable assertions (e.g., the cost of losing information) should be considered. Fourth, the certainty information on an assertion, given by the information extraction/co-reference process, can be used as the cost of losing information (called the *removal cost*), because deleting a more certain assertion generally loses more informa-

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

tion. Fifth, the collective removal cost of a set of assertions can be approximated by the sum of removal costs in the set.

Therefore, we in this paper address computing a *minimum cost diagnosis* for an inconsistent ontology KB, i.e. a subset of removable assertions whose removal turns KB consistent and in which the sum of removal costs is minimum. We show that, unless P=NP, the problem of finding a minimum cost diagnosis for a DL-Lite ontology is insolvable in polynomial time (PTIME) w.r.t. data complexity, i.e. the complexity measured in the size of the ABox only. Note that DL-Lite is a fairly inexpressive DL language such that the consistency checking problem for DL-Lite ontologies is in PTIME in the size of the ontology [2]. This complexity result implies that the problem of finding minimum cost diagnoses is in general intractable. In spite of that, we develop a feasible computational method for more general (i.e. $\mathcal{SHIQ}$) ontologies. It transforms a $\mathcal{SHIQ}$ ontology to a set of disjoint propositional programs by applying an existing transformation method (from $\mathcal{SHIQ}$ to disjunctive datalog) [12, 21] and new grounding and partitioning techniques. Thus our method reduces the problem of finding a minimum cost diagnosis into a set of independent subproblems. Each such subproblem computes an optimal model and is solvable in $O(\log_2 n)$ calls to a satisfiability (SAT) solver, by assuming that removal costs have been scaled to positive integers polynomial in n the number of removable assertions.

We implement our method and experiment on several originally consistent, real/benchmark ontologies. Each test ontology has over thousands of assertions. We implement a tool to inject conflicts into a consistent ontology, where a conflict, caused by several inserted assertions, violates a functional restriction or a disjointness constraint. Experimental results show that, even when all assertions are assumed removable, our method can handle all the test ontologies with injected conflicts. Especially, our method scales well on the extended benchmark ontologies with increasing number (from 1000) of conflicts.

2. RELATED WORK

There are some works that address repairing DL-based ontologies. Kalyanpur *et al.* [13] extended Reiter's Hitting Set Tree (HST) algorithm [24] to compute a minimum-rank hitting set, which is a subset of axioms that need to be removed/fixed to correct an unsatisfiable concept, such that the sum of axiom ranks in the subset is minimum. The notion of minimum-rank hitting set is similar to that of minimum cost diagnosis, except that the former is on axioms while the latter is on assertions. Schlobach [26] applied Reiter's HST algorithm to compute a minimal subset of axioms that need to be removed/fixed to correct an unsatisfiable concept or an incoherent TBox. The above methods require all minimal conflict sets be computed beforehand, where a *minimal conflict set* is a minimal subset of axioms responsible for the unwanted consequence. Though the above methods can work with ABoxes as well (by viewing assertions as axioms), it is impractical to adapt them to computing minimum cost diagnoses. First, the problem of finding a minimum hitting set from minimal conflict sets is intrinsically intractable [15]. Second, though there exist efficient methods for computing a minimal conflict set (e.g., [27, 14, 20]), computing all minimal conflict sets is still hard because the number of minimal conflict sets can be exponential in the number of assertions, as shown in the following example.

Example 1. Let the intensional part consist of two axioms $A \sqsubseteq \forall P.\neg A \sqcap \forall Q.\neg A$ and $\neg A \sqsubseteq \forall P.A \sqcap \forall Q.A$, and the ABox be $\{A(a_1), P(a_n, a_1), Q(a_n, a_1)\} \cup \{P(a_i, a_{i+1}), Q(a_i, a_{i+1}) \mid 1 \leq i \leq n-1\}$, where n is an odd number. Then the ontology is inconsistent, because $\neg A(a_1)$ is one of its consequences but conflicts with $A(a_1)$. The minimal conflict sets over the ABox are of the form $\{A(a_1), U_1(a_1, a_2), \ldots, U_{n-1}(a_{n-1}, a_n), U_n(a_n, a_1)\}$, where U_i is either P or Q. So the number of minimal conflict sets is 2^n. $\square$

Hence, a method that computes minimum cost diagnoses from minimal conflict sets may work in exponential time and exponential space w.r.t. data complexity (e.g., when it handles an ontology that is the union of the ontology in the above example and the ontology given in the proof of Theorem 2). In contrast, our method works in exponential time (more precisely, in logarithmic calls to an NP oracle) and polynomial space w.r.t. data complexity.

There exist some heuristics-based methods for repairing DL-based ontologies. Schlobach [25] proposed an approximate approach to computing a subset of axioms whose removal corrects an unsatisfiable concept or an incoherent TBox. Dolby *et al.* [4] exploited summarization and refinement techniques to compute a subset of assertions whose removal turns an inconsistent ontology consistent. Their proposed methods, however, cannot guarantee minimality for the set of removed axioms/assertions.

There also exist some methods for revising problematic axioms (e.g., [19, 13, 16, 23]). But they cannot be adapted to revising assertions, because assertions are assumed atomic in our work. We only consider the deletion of assertions.

As for dealing with inconsistency in DL-based ontologies, there is another approach that simply avoids/tolerates the inconsistency and applies a non-standard reasoning method to obtain meaningful answers (e.g., [11, 17]). We can also adapt our method to this approach, by defining a *consistent consequence* of an inconsistent ontology as a consequence invariant under all minimum cost diagnoses. This is out of the scope of this paper and is not discussed here.

3. PRELIMINARIES

3.1 SHIQ and DL-Lite

The $\mathcal{SHIQ}$ description logic [10] is a syntactic variant of OWL DL [22] without nominals and concrete domain specifications, but allowing qualified number restrictions.

A $\mathcal{SHIQ}$ RBox $KB_{\mathcal{R}}$ is a finite set of *transitivity axioms* $Trans(R)$ and *role inclusion axioms* $R \sqsubseteq S$, where R and S are roles. Let $\sqsubseteq^*$ be the reflexive transitive closure of $\{R \sqsubseteq S, Inv(R) \sqsubseteq Inv(S) \mid R \sqsubseteq S \in KB_{\mathcal{R}}\}$, where $Inv(R) = R^-$ and $Inv(R^-) = R$ for a role R. A role S is *simple* if there is no role R such that $R \sqsubseteq^* S$ and either $Trans(R) \in KB_{\mathcal{R}}$ or $Trans(Inv(R)) \in KB_{\mathcal{R}}$. The set of $\mathcal{SHIQ}$ concepts is the smallest set containing $\top$, $\bot$, A, $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\leq_n S.C$ and $\geq_n S.C$, where A is a concept name (i.e. an *atomic concept*), C and D $\mathcal{SHIQ}$ concepts, R a role, S a simple role, and n a positive integer. A $\mathcal{SHIQ}$ TBox $KB_{\mathcal{T}}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are $\mathcal{SHIQ}$ concepts. A $\mathcal{SHIQ}$ ABox $KB_{\mathcal{A}}$ is a set of *concept assertions* $C(a)$, *role assertions* $R(a, b)$, *equality assertions* $a \approx b$ and *inequality assertions* $a \not\approx b$, where C is a $\mathcal{SHIQ}$ concept, R a role, and a and b individuals. A $\mathcal{SHIQ}$ ontology KB is a triple $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$, where

$KB_{\mathcal{R}}$ is an RBox, $KB_{\mathcal{T}}$ a TBox, and $KB_{\mathcal{A}}$ an ABox. In this paper, by KB we simply denote $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$ if there is no confusion.

DL-Lite [2] is a sub-language of $\mathcal{SHIQ}$. The set of DL-Lite concepts is the smallest set containing A, $\exists R$, $\exists R^-$, $\neg B$ and $C_1 \sqcap C_2$, where A is a concept name, R a role name, B a *basic concept* (i.e. a concept of the form A, $\exists R$ or $\exists R^-$), and C_1 and C_2 DL-Lite concepts. $\exists R$ is actually an unqualified existential restriction $\exists R.\top$. A DL-Lite ontology $KB = (KB_{\mathcal{T}}, KB_{\mathcal{A}})$ consists of a TBox $KB_{\mathcal{T}}$ and an ABox $KB_{\mathcal{A}}$. $KB_{\mathcal{T}}$ is a finite set of *inclusion axioms* $B \sqsubseteq C$ and *functionality axioms* (func R) and (func R^-), where B is a basic concept, C a DL-Lite concept and R a role. $KB_{\mathcal{A}}$ is a set of *concept assertions* $B(a)$ and *role assertions* $R(a, b)$, where B is a basic concept, R a role, and a and b individuals.

The semantics of a $\mathcal{SHIQ}$ ontology KB is given by a mapping π that translates KB into first-order logic. Due to space limitation, we refer readers to [12] for the definition of π. KB is said to be *consistent/satisfiable* if there exists a first-order model of $\pi(KB)$. The semantics of a DL-Lite ontology KB can still be given by the same mapping π, because a functionality axiom (func R) is a syntactic variant of $\top \sqsubseteq \leq_1 R.\top$. Note that the *unique name assumption* (UNA) [1] on individuals is applied in DL-Lite but not in $\mathcal{SHIQ}$. UNA can be explicitly axiomatized by appending to the ABox all inequality assertions $a \not\approx b$ for any two individuals a and b that have different URIs.

In our work we assume that all concept assertions are attached to atomic concepts only, due to the following reasons. First, to the best of our knowledge, most of the existing OWL ontologies in the RDF/XML syntax (cf. `http://www.w3.org/TR/owl-ref`) have no non-atomic concept assertions. Second, a non-atomic concept assertion $C(a)$ can be reduced to an atomic one by replacing $C(a)$ with $Q(a)$ and appending $\{C \sqsubseteq Q, Q \sqsubseteq C\}$ to the TBox, where Q is a new atomic concept.

3.2 Disjunctive Datalog

A *disjunctive datalog program with equality* [6] P is a finite set of rules without function symbols of the form $A_1 \vee \ldots \vee A_n \leftarrow B_1, \ldots, B_m$ (where A_i and B_i are atoms). Each rule must be *safe*, i.e., each variable occurring in a head atom A_i must occur in some body atom B_j. For a rule r, the set of head atoms is denoted by $head(r)$, whereas the set of body atoms is denoted by $body(r)$. A rule r is called a *constraint* if $|head(r)| = 0$; a *fact* if $|body(r)| = 0$. An atom is called *negated* if it leads with negation-as-failure. Typical definitions of a disjunctive datalog program, such as [6], allow negated atoms in the body. In our work, negated atoms cannot occur in a transformed program that we consider, so we omit negation-as-failure from the definitions. Disjunctive datalog programs without negation-as-failure are often called *positive programs.*

The set of all ground instances of rules in P is denoted by $ground(P)$. An *interpretation* M of P is a subset of ground atoms in the Herbrand base of P. An interpretation M is called a *model* of P if (i) $body(r) \subseteq M$ implies $head(r) \cap M \neq \emptyset$ for each rule $r \in ground(P)$, and (ii) all atoms from M with the equality predicate $\approx$ yield a congruence relation, i.e. a relation that is reflexive, symmetric, transitive, and $T(a_1, \ldots, a_i, \ldots, a_n) \in M$ and $a_i \approx b_i \in M$ imply $T(a_1, \ldots, b_i, \ldots, a_n) \in M$ for each predicate symbol T in P. P is said to be *satisfiable* if it has a model.

3.3 Reducing SHIQ to Disjunctive Datalog

Since $\mathcal{SHIQ}$ is a subset of first-order logic, $\mathcal{SHIQ}$ axioms can first be translated into logical formulas, then into clausal form. The resulting clauses can be represented as disjunctive rules without negation-as-failure. However, due to possible skolemization steps in the clausal form translation, the resulting rules may contain function symbols. Standard logic program engines, however, may not terminate in the presence of function symbols. To cope with this problem, Hustadt *et al.* [12, 21] developed the *KAON2 transformation method* to get rid of function symbols without losing ABox consequences.

The method reduces a $\mathcal{SHIQ}$ ontology KB to a positive disjunctive datalog program $DD(KB) = \Gamma(KB_{\mathcal{R}}, KB_{\mathcal{T}}) \cup \Xi(KB_{\mathcal{A}}) \cup \Delta(KB)$. $\Gamma(KB_{\mathcal{R}}, KB_{\mathcal{T}})$ is a set of disjunctive datalog rules computed from the intensional part of KB by translating $\mathcal{SHIQ}$ axioms into clauses, adding logical consequences, and translating clauses into disjunctive datalog rules. $\Xi(KB_{\mathcal{A}})$ is a set of facts translated from $KB_{\mathcal{A}}$, where each inequality assertion (of the form $a \not\approx b$) is translated into a ground constraint (of the from $\leftarrow a \approx b$), and other assertions are directly translated into ground facts. $\Delta(KB)$ is a set of facts of the form $HU(a), HU(a_f)$ and $S_f(a, a_f)$, which are introduced to remove function symbols and instantiated for each individual a occurring in KB and each function symbol f.

THEOREM 1 ([21]). *For KB a $\mathcal{SHIQ}$ ontology, KB is unsatisfiable if and only if $DD(KB)$ is unsatisfiable.* $\square$

4. MINIMUM COST DIAGNOSIS

Given a possibly inconsistent $\mathcal{SHIQ}$ ontology KB in which some assertions are removable and assigned removal costs, our goal is to find a subset of removable assertions whose removal turns KB consistent and in which the sum of removal costs is minimum. Such subset is called a *minimum cost diagnosis*, formally defined below.

Definition 1. Let KB be a possibly inconsistent $\mathcal{SHIQ}$ ontology and $R_{KB} \subseteq KB_{\mathcal{A}}$ a set of removable assertions such that each assertion $\alpha \in R_{KB}$ is given a *removal cost* $c(\alpha) > 0$. Then, a subset of assertions $R \subseteq R_{KB}$ is called a *diagnosis* for KB w.r.t. R_{KB} if $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}} \setminus R)$ is consistent. A diagnosis R is called a *minimum cost diagnosis* for KB w.r.t. R_{KB} if there is no diagnosis R' for KB w.r.t. R_{KB} such that $\sum_{\alpha \in R'} c(\alpha) < \sum_{\alpha \in R} c(\alpha)$. R is simply called a diagnosis/minimum cost diagnosis if KB and R_{KB} are clear from the context.

We consider the time complexity for finding a minimum cost diagnosis. Complexity results in this paper refer to *data complexity*, i.e. the complexity measured as a function of the number of assertions in the ontology. Theorem 2 shows that, unless P=NP, there is no polynomial time algorithm for finding a minimum cost diagnosis for a DL-Lite ontology KB w.r.t. $KB_{\mathcal{A}}$. It implies that the problem of finding minimum cost diagnoses for $\mathcal{SHIQ}$ ontologies is in general intractable.

THEOREM 2. *Given a positive integer k and a possibly inconsistent DL-Lite ontology $KB = (KB_{\mathcal{T}}, KB_{\mathcal{A}})$ where each assertion $\alpha \in KB_{\mathcal{A}}$ is given a removal cost $c(\alpha) = 1$, deciding if there is a diagnosis R for KB w.r.t. $KB_{\mathcal{A}}$ such that $\sum_{\alpha \in R} c(\alpha) \leq k$ is NP-hard w.r.t. data complexity.*

PROOF. Given an arbitrary instance I of the SAT problem, we transform it into an instance I' of the given decision problem. Let I be the formula $f = C_1 \wedge \ldots \wedge C_m$ with m clauses and n boolean variables $x_1, \ldots, x_n$. We construct I' as follows. (1) $KB_{\mathcal{T}}$ consists of the following axioms:

$\exists T \sqsubseteq \neg\exists S$, $\exists T^- \sqsubseteq \neg\exists S^-$, (func T^-), (func S^-).

(2) For each boolean variable x_i in f, $KB_{\mathcal{A}}$ contains a corresponding constant a_i. (3) For each clause C_j containing n_j literals $l_{j,1}, \ldots, l_{j,n_j}$ whose corresponding constants in $KB_{\mathcal{A}}$, introduced in (2), are $a_{j,1}, \ldots, a_{j,n_j}$ respectively, $KB_{\mathcal{A}}$ contains a corresponding constant c_j for C_j and n_j assertions $U(a_{j,1}, c_j)$, ..., $U(a_{j,n_j}, c_j)$, where $U(a_{j,q}, c_j)$ is $T(a_{j,q}, c_j)$ if $l_{j,q}$ is positive, or $S(a_{j,q}, c_j)$ otherwise. (4) Let $k = \sum_{j=1}^{m}(n_j - 1)$.

Now we prove that f is satisfiable if and only if there is a diagnosis R for $KB = (KB_{\mathcal{T}}, KB_{\mathcal{A}})$ w.r.t. $KB_{\mathcal{A}}$ such that $\sum_{a \in R} c(a) \le k$, i.e., $|R| \le k$. ($\Rightarrow$) Since f is satisfiable, for each clause C_j there is a literal l_{j,t_j} assigned true. We append to R all assertions in $KB_{\mathcal{A}}$ of the form $U(a_{j,p}, c_j)$ ($p \neq t_j$), where $U(a_{j,p}, c_j)$ is $T(a_{j,p}, c_j)$ if $l_{j,p}$ is positive, or $S(a_{j,p}, c_j)$ otherwise. Clearly, R is a diagnosis for KB w.r.t. $KB_{\mathcal{A}}$ such that $|R| \le k$. ($\Leftarrow$) Suppose R is a diagnosis for KB w.r.t. $KB_{\mathcal{A}}$ such that $|R| \le k$. It is not hard to see that any set of assertions of the form $U(a_{j,q}, c_j)$ (U is either T or S) must have exactly $n_j - 1$ assertions in R and one in $KB_{\mathcal{A}} \setminus R$. To see that f is satisfiable, for each $U(a_{j,t_j}, c_j) \in KB_{\mathcal{A}} \setminus R$ ($j = 1, \ldots, m$), if U is T, we assign x_{j,t_j} (i.e. the corresponding variable of a_{j,t_j} in f) true; otherwise we assign x_{j,t_j} false. The above (partial) assignment on $\{x_1, \ldots, x_n\}$ is consistent and ensures $l_{j,t_j} =$ true for all $j = 1, \ldots, m$. Thus f is satisfiable.

Since the construction of KB is accomplished in PTIME and the SAT problem is NP-complete, and since $KB_{\mathcal{T}}$ has a fixed size, this theorem follows. □

5. COMPUTING MINIMUM COST DIAGNOSES

As analyzed in related work, a method that computes minimum cost diagnoses based on minimal conflict sets is impractical, because it may require both exponential time and exponential space. We thus consider methods that need not compute minimal conflict sets. A naïve method is the black-box method, which searches minimum cost diagnoses over all subsets of removal assertions by applying a DL reasoner to check diagnoses. However, the black-box method cannot compute a minimum cost diagnosis for a DL-Lite ontology in polynomial calls to a DL reasoner, otherwise a minimum cost diagnosis can be computed in PTIME w.r.t. data complexity, contradicting Theorem 2. In order to find a practical computational method, we consider transforming the input ontology KB into a positive program Π such that for any subset S of R_{KB} the set of removable assertions, $KB \setminus S$ is consistent if and only if $\Pi \cup \{\text{assign}(\alpha^-) = 1 \mid \alpha \in S\} \cup \{\text{assign}(\alpha^-) = 0 \mid \alpha \in R_{KB} \setminus S\}$ is satisfiable, where α^- is a fresh ground atom corresponding to ground atom α in Π, and assign(β) denotes the 0-1 truth value of β. Then, a minimum cost diagnosis corresponds to a valuation of X such that $\sum_{\alpha^- \in X} c(\alpha) \cdot \text{assign}(\alpha^-)$ is minimum and Π is satisfiable, where $X = \{\alpha^- \mid \alpha \in R_{KB}\}$.

To find such a valuation of X, we need to handle *pseudo-boolean constraints* (PB-constraints) of the form $\sum_i c_i x_i \le d$ with constants $c_i, d \in \mathbf{Z}$ and variables $x_i \in \{0,1\}$, or a linear optimization function of the form `minimize` $\sum_i c_i x_i$ with constants $c_i \in \mathbf{Z}$ and variables $x_i \in \{0,1\}$, where $\mathbf{Z}$ denotes the integer domain. The SAT problems with PB-constraints and linear optimization functions are well studied in the SAT community (cf. `http://www.cril.univ-artois.fr/PB07/`). A SAT problem with linear optimization functions can be translated into a set of SAT problems with PB-constraints. A SAT problem with PB-constraints can be either solved by standard SAT solvers after translating PB-constraints to SAT clauses [5], or solved by extended SAT solvers that support PB-constraints natively (e.g., PUEBLO [28]).

Now, the remaining problems are how to transform a $\mathcal{SHIQ}$ ontology to the intended positive program and how to efficiently compute minimum cost diagnoses. We address these problems in the following subsections.

5.1 Constructing a Repair Program

Given a possibly inconsistent $\mathcal{SHIQ}$ ontology KB, we first employ the KAON2 transformation method [12, 21], described in Preliminaries, to reduce KB to a disjunctive datalog program $DD(KB) = \Gamma(KB_{\mathcal{R}}, KB_{\mathcal{T}}) \cup \Xi(KB_{\mathcal{A}}) \cup \Delta(KB)$, but introduce a special treatment. The original KAON2 transformation method allows equality atoms (of the form $X \approx Y$ or $a \approx b$, where X, Y denote variables and a, b denote constants) to occur in rule bodies in $DD(KB)$ while disallows inequality atoms (of the form $X \not\approx Y$ or $a \not\approx b$) to occur in $DD(KB)$. To handle inequality assertions in a similar way as other assertions, we first move equality atoms ($X \approx Y$ or $a \approx b$) in any rule body in $DD(KB)$ to the corresponding rule head and replace them with inequality atoms ($X \not\approx Y$ or $a \not\approx b$), then append to $DD(KB)$ a constraint $\leftarrow X \approx Y, X \not\approx Y$ (written $R_{\not\approx}$), so that $\Xi(KB_{\mathcal{A}})$ is simplified to a direct translation from assertions in $KB_{\mathcal{A}}$ to ground facts in $DD(KB)$. Having such treatment we simply denote $\Xi(KB_{\mathcal{A}})$ as $KB_{\mathcal{A}}$. The modified rules in $DD(KB)$ are still safe due to the restricted form of the original rules that have equality atoms in the body. In essence, our treatment views an inequality atom as an ordinary one and does not impact the satisfiability of $DD(KB)$. Then, we convert $DD(KB)$ to a *repair program* $\mathcal{R}(KB)$ defined below. Intuitively, the *decision atom* α^- is introduced to weaken KB, so that $\alpha^- =$ true (resp. $\alpha^- =$ false) implies that α is removed from (resp. kept in) KB. Note that decision atoms in $\mathcal{R}(KB)$ are treated as nullary ground atoms.

Definition 2. For KB a possibly inconsistent $\mathcal{SHIQ}$ ontology and $R_{KB} \subseteq KB_{\mathcal{A}}$ a set of removable assertions such that each assertion $\alpha \in R_{KB}$ is given a removal cost $c(\alpha) > 0$, a *repair program* of KB w.r.t. R_{KB}, written $\mathcal{R}(KB)$, is a disjunctive datalog program converted from $DD(KB)$ as follows: for each assertion $\alpha \in R_{KB}$, we introduce a corresponding *decision atom* α^- and give it a cost $c(\alpha^-) = c(\alpha)$, then replace the ground fact α in $DD(KB)$ with $\alpha \vee \alpha^-$. We simply call $\mathcal{R}(KB)$ a repair program if R_{KB} is clear from the context.

Example 2. Let A, E, H, P, S, T, me and pa abbreviate $Artificer$, $Engineer$, $Human$, $Professor$, $Student$, $Teacher$, $mentor$ and $parent$ respectively. Given a $\mathcal{SHIQ}$ ontology $KB = (\emptyset, KB_{\mathcal{T}}, KB_{\mathcal{A}})$, where $KB_{\mathcal{T}} = \{S \sqsubseteq \le_1 me \sqcap \exists me.P \sqcap H,\ H \sqsubseteq \forall pa.H,\ P \sqsubseteq E,\ \exists me.E \sqsubseteq \neg A,\ E \sqsubseteq \neg T\}$ and $KB_{\mathcal{A}} = \{S(s_1), S(s_2), me(s_1, t_1), me(s_1, t_2), A(s_2), T(t_1), T(t_2), T(p_1), E(p_2), pa(s_1, p_1), pa(s_2, p_1), t_1 \not\approx t_2,\ p_1 \approx p_2\}$, and a set of removable assertions $R_{KB} =$

$\{S(s_2),\ A(s_2),\ T(t_1),\ T(t_2),\ E(p_2),\ t_1 \not\approx t_2,\ p_1 \approx p_2\}$ such that $c(\alpha) = 1$ for each assertion $\alpha \in R_{KB}$, we construct the repair program $\mathcal{R}(KB)$ as follows.

First, by applying the KAON2 transformation method with our special treatment, we reduce KB to $DD(KB) = \Gamma(KB_{\mathcal{R}}, KB_{\mathcal{T}}) \cup \{R_{\not\approx}\} \cup KB_{\mathcal{A}} \cup \Delta(KB)$, where $\Delta(KB) = \{S_f(s_1, s_{1f}), S_f(s_2, s_{2f}), S_f(t_1, t_{1f}), S_f(t_2, t_{2f}), S_f(p_1, p_{1f})\}$ and $\Gamma(KB_{\mathcal{R}}, KB_{\mathcal{T}}) = \{R_1, \ldots, R_9\}$ as given below.

R_1: $Y_1 \approx Y_2 \leftarrow S(X), me(X, Y_1), me(X, Y_2).$
R_2: $P(X_f) \leftarrow S(X), S_f(X, X_f).$
R_3: $me(X, X_f) \leftarrow S(X), S_f(X, X_f).$
R_4: $H(X) \leftarrow S(X).$
R_5: $H(Y) \leftarrow H(X), pa(X, Y).$
R_6: $E(X) \leftarrow P(X).$
R_7: $\leftarrow A(X), me(X, Y), E(Y).$
R_8: $\leftarrow T(X), E(X).$
R_9: $\leftarrow A(X), S(X).$

Then, by introducing decision atoms and converting ground facts in $DD(KB)$, we obtain $\mathcal{R}(KB) = \{R_1, \ldots, R_9, R_{\not\approx}\} \cup \Delta(KB) \cup \{S(s_1),\ me(s_1, t_1),\ me(s_1, t_2),\ T(p_1),\ pa(s_1, p_1),\ pa(s_2, p_1),\ S(s_2) \vee S(s_2)^-,\ A(s_2) \vee A(s_2)^-,\ T(t_1) \vee T(t_1)^-,\ T(t_2) \vee T(t_2)^-,\ E(p_2) \vee E(p_2)^-,\ (t_1 \not\approx t_2) \vee (t_1 \not\approx t_2)^-,\ (p_1 \approx p_2) \vee (p_1 \approx p_2)^-\}$. □

There exists a correspondence between minimum cost diagnoses for KB w.r.t. R_{KB} and X-MC models of $\mathcal{R}(KB)$, where $X = \{\alpha^- \mid \alpha \in R_{KB}\}$ (see Theorem 3). A model M of a positive program P is called an *X-MC model* of P if there is no model M' of P such that $\sum_{\beta \in M' \cap X} c(\beta) < \sum_{\beta \in M \cap X} c(\beta)$, where X is a set of ground atoms and c is a predefined cost function over X.

THEOREM 3. *Let KB be a $\mathcal{SHIQ}$ ontology, $R_{KB} \subseteq KB_{\mathcal{A}}$ a set of removable assertions such that each assertion $\alpha \in R_{KB}$ is given a removal cost $c(\alpha) > 0$, $\mathcal{R}(KB)$ a repair program of KB w.r.t. R_{KB}, and $X = \{\alpha^- \mid \alpha \in R_{KB}\}$.*

(Soundness) For each X-MC model M of $\mathcal{R}(KB)$, $\{\alpha \mid \alpha^- \in M\}$ is a minimum cost diagnosis for KB w.r.t. R_{KB};

(Completeness) For each minimum cost diagnosis R for KB w.r.t. R_{KB}, there exists an X-MC model M of $\mathcal{R}(KB)$ such that $R = \{\alpha \mid \alpha^- \in M\}$.

PROOF SKETCH. (Soundness) Let M be an X-MC model of $\mathcal{R}(KB)$, $R = \{\alpha \mid \alpha^- \in M\}$ and $M' = M \setminus \{\alpha^- | \alpha \in R\}$. It can be shown that M' is a model of $DD(KB) \setminus R$. By Theorem 1, R is diagnosis of KB w.r.t. R_{KB}. Further, R must be a minimum cost diagnoses, otherwise it can be shown that there exists a model M'' of $\mathcal{R}(KB)$ s.t. $\sum_{\alpha^- \in M'' \cap X} c(\alpha^-) < \sum_{\alpha^- \in M \cap X} c(\alpha^-)$.

(Completeness) Let R be a minimum cost diagnosis for KB w.r.t. R_{KB}. By Theorem 1, $DD(KB) \setminus R$ is satisfiable and thus has a model, say M. It can be shown that $M' = M \cup \{\alpha^- \mid \alpha \in R\}$ is a model of $\mathcal{R}(KB)$. Further, M' must be an X-MC model of $\mathcal{R}(KB)$, otherwise it can be shown that there exists a diagnosis R' for KB w.r.t. R_{KB} s.t. $\sum_{\alpha \in R'} c(\alpha) < \sum_{\alpha \in R} c(\alpha)$. □

5.2 Computing X-MC Models

By Theorem 3, the problem of finding minimum cost diagnoses for KB w.r.t. R_{KB} is reduced to the problem of computing X-MC models of $\mathcal{R}(KB)$, which can be realized by applying SAT solvers. However, SAT solvers take a positive propositional program as input and do not distinguish equality atoms from other atoms. To treat the equality predicate $\approx$, which is interpreted as a congruence relation, as an ordinary predicate, we use a well-known transformation from [8]. For a disjunctive datalog program P, let $P_{\approx}$ denote the program consisting of the rules stating that the equality predicate is *reflexive*, *symmetric* and *transitive*, and the *replacement rules* given below, instantiated for each predicate T in P (excluding $\approx$) and each position i. Note that the reflexive rule is not safe and is instead represented as a set of ground facts of the form $a \approx a$, instantiated for each constant a in P. Then, appending $P_{\approx}$ to P allows to treat $\approx$ as an ordinary predicate.

$$T(X_1, \ldots, Y_i, \ldots, X_n) \leftarrow X_i \approx Y_i, T(X_1, \ldots, X_i, \ldots, X_n).$$

For the input issue, we need to ground $\mathcal{R}(KB)$ before applying SAT solvers. A well-known grounding technique is *intelligent grounding* (IG) [7], which only applies to equality-free disjunctive datalog programs. That is, if the equality predicate $\approx$ is present in a disjunctive datalog program P, we must append $P_{\approx}$ to P before grounding P using the IG technique. The IG technique has been implemented in a disjunctive datalog engine DLV [18], but the current implementation cannot handle large disjunctive datalog programs due to memory limitation[1], especially when the equality predicate is present. On the other hand, current implementations of SAT solvers lack scalability for large propositional programs. To address these problems, we develop two disk-based algorithms for grounding $\mathcal{R}(KB)$ to $\Pi(KB)$ and for partitioning $\Pi(KB)$ to disjoint subprograms respectively, so that the computation of minimum cost diagnoses can be separately performed over each subprogram.

Algorithm 1 is our algorithm for grounding a repair program P. By M_{def} we denote the unique minimal model of the *definite fragment* of P, i.e. $\{R \in P \mid |head(R)| = 1\}$. $\mathcal{C}$ is actually the set of congruence classes $\{C_1, \ldots, C_m\}$ occurring in P, where $C_i = \{b \mid a \approx b \in M_{def}\}$ for an arbitrary constant a occurring in some equality atom in M_{def} that is not of the form $a \approx a$. $f_c(a, \mathcal{C})$ denotes the congruence class in $\mathcal{C}$ that contains constant a. $\min_c(C)$ denotes the constant $a \in C$ having the smallest value in $\{occ(a) \mid a \in C\}$, where $occ(a)$ is the occurrence order of a in P. $\mathcal{D}_{def}$ is actually a set of non-equality atoms in M_{def} such that for each non-equality atom $T(a_1, \ldots, a_k) \in M_{def}$, there exists a unique ground atom $T(b_1, \ldots, b_k) \in \mathcal{D}_{def}$ such that for each $i = 1, \ldots, k$, a_i and b_i are either the same or together in some $C \in \mathcal{C}$. $\mathcal{D}$ is the set of ground atoms occurring in the grounded program Π.

Let S and S' be two sets of ground atoms. $S_{\approx}$ denotes the subset of S consisting of all equality atoms in S; $S_{\setminus\approx}$ denotes $S \setminus S_{\approx}$. For a rule R, the function `GetSubstitutes`(R, S, S') returns the set of all ground substitutes σ such that $body(R\sigma) \subseteq S$, $head(R\sigma)_{\setminus\approx} \cap S' = \emptyset$ and $head(R\sigma)_{\approx}$ does not contain equality atoms of the form $a \approx a$. The function `Rewrite`$(S, \mathcal{C})$ rewrites all constants a in S such that $f_c(a, \mathcal{C})$ exists to $min_c(f_c(a, \mathcal{C}))$, and returns the modified S.

The algorithm consists of three steps. Step 1 (lines 1–13) computes M_{def} in a standard iterative manner, but represents M_{def} as $\mathcal{D}_{def}$ and $\mathcal{C}$. Step 2 (lines 14–16) rewrites the constants occurring in disjunctive ground facts (of the form $\alpha \vee \alpha^-$) in P, because some constants occurring in α

[1]The current implementation with DB support, DLV^{DB} (`http://www.mat.unical.it/terracina/dlvdb/`), does not work with DBs if the input program has disjunctions.

are represented by other constants in step 1. At a word, in step 1 and step 2, all constants in a congruence class are replaced with a representative constant, so as to reduce the number of instantiated rules in step 3. Step 3 (lines 17–26) grounds P^+, i.e. $P \cup P_\approx$ excluding the definite ground facts, in a standard iterative manner based on M_{def}. Each instantiated rule r such that $head(r) \cap M_{def} \neq \emptyset$ is ignored (line 22), because r has no impact on computing models of P. If $a \not\approx b \in \mathcal{D}_{def}$, the equality atom $a \approx b$ in an instantiated rule head is not appended to $\mathcal{D}$ (line 24), because it cannot occur in any model of P. The ground atoms in M_{def} are removed from the body of any instantiated rule (lines 25–26), because they are in every model of P. Note that the function `GetSubstitutes` can be realized by applying a SQL engine and its results can be stored in disk, so the algorithm is disk-based. In what follows, by $\Pi(KB)$ we denote the grounded repair program returned by `Ground`$(\mathcal{R}(KB))$.

Algorithm 1. `Ground`(P)

```
Input: A repair program $P$.
Output: A set $\mathcal{C}$ of sets of constants and a positive propositional
    program $\Pi$.
1.  $\mathcal{C} := \emptyset$; $\mathcal{D}_{def} := \emptyset$; $\mathcal{D}'_{def} := \{\bot\}$; // to enforce $\mathcal{D}_{def} \neq \mathcal{D}'_{def}$
2.  while $\mathcal{D}_{def} \neq \mathcal{D}'_{def}$ do
3.    $\mathcal{D}'_{def} := \mathcal{D}_{def}$;
4.    for each rule $R \in P$ s.t. $|head(R)| = 1$ do
5.      $\Theta :=$ GetSubstitutes($R$, $\mathcal{D}_{def}$, $\mathcal{D}_{def}$);
6.      for each $\sigma$ sequentially retrieved from $\Theta$ do
7.        $\mathcal{D}_{def} := \mathcal{D}_{def} \cup head(R\sigma)_{\setminus\approx}$;
8.        if $head(R\sigma)_\approx = \{a \approx b\}$ for some constants $a$ and $b$
    that are not together in some $C \in \mathcal{C}$ then
9.          if $f_c(a, \mathcal{C})$ does not exist then Set $f_c(a, \mathcal{C})$ as $\{a\}$;
10.         if $f_c(b, \mathcal{C})$ does not exist then Set $f_c(b, \mathcal{C})$ as $\{b\}$;
11.         $C := f_c(a, \mathcal{C}) \cup f_c(b, \mathcal{C})$;
12.         $\mathcal{C} := (\mathcal{C} \setminus \{f_c(a, \mathcal{C}), f_c(b, \mathcal{C})\}) \cup \{C\}$;
13.     $\mathcal{D}_{def} :=$ Rewrite($\mathcal{D}_{def}, \mathcal{C}$); // executed once for $\Theta$
14. for each disjunctive ground fact $\alpha \vee \alpha^-$ in $P$ do
15.   for each constant $a$ in $\alpha$ (or $\alpha^-$) s.t. $f_c(a, \mathcal{C})$ exists do
16.     Rewrite $a$ in $\alpha$ (or $\alpha^-$) to $min_c(f_c(a, \mathcal{C}))$;
17. $P^+ := \{R \in P \cup P_\approx \mid |head(R)| > 1$ or $R$ is non-ground$\}$;
18. $\Pi := \emptyset$; $\mathcal{D} := \mathcal{D}_{def} \cup \{a \approx a \mid a$ occurs in $P\}$; $\mathcal{D}' := \emptyset$;
19. while $\mathcal{D} \neq \mathcal{D}'$ do
20.   $\mathcal{D}' := \mathcal{D}$;
21.   for each rule $R \in P^+$ do
22.     $\Theta :=$ GetSubstitutes($R$, $\mathcal{D}$, $\mathcal{D}_{def}$);
23.     for each $\sigma$ sequentially retrieved from $\Theta$ do
24.       $\mathcal{D} := \mathcal{D} \cup head(R\sigma)_{\setminus\approx} \cup \{a \approx b \in head(R\sigma)_\approx \mid a \not\approx
    b \notin \mathcal{D}_{def}\}$;
25.       $B := body(R\sigma) \setminus (\mathcal{D}_{def} \cup \{a \approx a \mid a$ occurs in $R\sigma\})$;
26.       $\Pi := \Pi \cup \{\bigvee head(R\sigma) \leftarrow \bigwedge B\}$;
27. return $(\mathcal{C}, \Pi)$;
```

Example 3. Continue with Example 2. We now demonstrate how `Ground`$(\mathcal{R}(KB))$ works. In step 1, we compute the unique minimal model M_{def} of $\mathcal{R}(KB)_{def}$ in an iterative manner, obtaining $\mathcal{C} = \{\{t_1, t_2, s_{1f}\}\}$ and $\mathcal{D}_{def} = \{S(s_1), me(s_1,t_1), T(p_1), pa(s_1,p_1), pa(s_2,p_1), S_f(s_1,t_1), P(t_1), H(s_1), H(p_1), E(t_1), S_f(s_2,s_{2f}), S_f(t_1,t_{1f}), S_f(t_1,t_{2f}), S_f(p_1,p_{1f})\}$. In step 2, according to $\mathcal{C}$, we replace the set of disjunctive ground facts in $\mathcal{R}(KB)$ with $\{S(s_2) \vee S(s_2)^-, A(s_2) \vee A(s_2)^-, T(t_1) \vee T(t_1)^-, E(p_2) \vee E(p_2)^-, (t_1 \not\approx t_1) \vee (t_1 \not\approx t_1)^-, (p_1 \approx p_2) \vee (p_1 \approx p_2)^-\}$. In step 3, we ground $\mathcal{R}(KB) \cup \mathcal{R}(KB)_\approx$ (excluding the definite ground facts) in an iterative manner, obtaining a propositional program $\Pi(KB) = \{r_1, \ldots, r_{21}\}$, where r_{14} is instantiated from $\leftarrow X \approx Y, X \not\approx Y$ (i.e. $R_{\not\approx}$), $r_{15}, \ldots, r_{19}$ are instantiated from $\mathcal{R}(KB)_\approx$. Note that $p_2 \approx p_1$ has been rewritten to $p_1 \approx p_2$ in $\Pi(KB)$ so as to delete the tautological rules of the form $a \approx b \leftarrow a \approx b$ without impacting the computation of X-MC models of $\Pi(KB)$.

$r_1 : S(s_2) \vee S(s_2)^-.$ $\quad r_2 : A(s_2) \vee A(s_2)^-.$
$r_3 : (t_1 \not\approx t_1) \vee (t_1 \not\approx t_1)^-.$ $\quad r_4 : T(t_1) \vee T(t_1)^-.$
$r_5 : (p_1 \approx p_2) \vee (P_1 \not\approx p_2)^-.$ $\quad r_6 : E(p_2) \vee E(p_2)^-.$
$r_7 : P(s_{2f}) \leftarrow S(s_2).$ $\quad r_8 : me(s_2, s_{2f}) \leftarrow S(s_2).$
$r_9 : H(s_2) \leftarrow S(s_2).$ $\quad r_{10} : E(s_{2f}) \leftarrow P(s_{2f}).$
$r_{11} :\leftarrow A(s_2), me(s_2, s_{2f}), E(s_{2f}).$ $\quad r_{12} :\leftarrow T(t_1).$
$r_{13} :\leftarrow A(s_2), S(s_2).$ $\quad r_{14} :\leftarrow t_1 \not\approx t_1.$
$r_{15} : T(p_2) \leftarrow p_1 \approx p_2.$ $\quad r_{16} : E(p_2) \leftarrow p_1 \approx p_2, E(p_1).$
$r_{17} : pa(s_1, p_2) \leftarrow p_1 \approx p_2.$ $\quad r_{18} : E(p_1) \leftarrow p_1 \approx p_2, E(p_2).$
$r_{19} : pa(s_2, p_2) \leftarrow p_1 \approx p_2.$
$r_{20} :\leftarrow E(p_1).$ $\quad r_{21} :\leftarrow T(p_2), E(p_2).$ □

Algorithm 2. `Partition`(Π, X)

```
Input: A positive propositional program $\Pi$ and a set $X$ of ground
    atoms occurring in $\Pi$.
Output: A set of disjoint subprograms of $\Pi$.
1.  Set $map(\alpha)$ as 0 for all ground atoms $\alpha$ occurring in $\Pi$;
2.  Move constraints in $\Pi$ in front of other rules in $\Pi$; $k := 0$;
3.  repeat
4.    $merged :=$ false;
5.    for each rule $r$ sequentially retrieved from $\Pi$ s.t. $head(r) =
    \emptyset$ or $map(\alpha) > 0$ for all $\alpha \in head(r) \setminus X$ do
6.      for each $\alpha \in head(r) \cup body(r)$ s.t. $map(\alpha) = 0$ do
7.        $k := k + 1$; $map(\alpha) := k$;
8.      if $|map(r)| > 1$ then
9.        $merged :=$ true; $min_{id} := \min(map(r))$;
10.       for each $\alpha \in head(r) \cup body(r)$ do $map(\alpha) := min_{id}$;
11. until not $merged$;
12. for $i = 1, \ldots, k$ do
13.   $\Pi_i := \{r \in \Pi \mid \forall\alpha \in head(r) \cup body(r) : map(\alpha) = i\}$;
14. return $\{\Pi_i \neq \emptyset \mid 1 \leq i \leq k\}$;
```

Algorithm 2 is our algorithm for partitioning a positive propositional program Π based on a set X of ground atoms occurring in Π. The basic idea is to filter out rules that have no impact on $M \cap X$ when constructing an X-MC model M of Π and put together remaining rules that have common ground atoms to form disjoint subprograms. In the algorithm, each ground atom α occurring in Π is mapped to a partition identifier $map(\alpha)$. For a rule r, we use $map(r)$ to denote $\{map(\alpha) \mid \alpha \in head(r) \cup body(r)\}$. To simplify explanation, we call a rule $r \in \Pi$ *ready* if $head(r) = \emptyset$ or $map(\alpha) > 0$ for all $\alpha \in head(r) \setminus X$. Before a ground atom is detected in some ready rule, it is mapped to 0 (line 1). To process ready rules as early as possible, constraints (which are ready rules) are moved in front of other rules in Π (line 2). Then, $\{map(\alpha) \mid \alpha$ occurs in $\Pi\}$ is adjusted in an iterative manner until $\{map(r) \mid r \in \Pi\}$ reaches a fixpoint (lines 3–11). Each ground atom α first detected in ready rules is initially mapped to a unique partition identifier (lines 6–7). All ground atoms in a ready rule r are mapped to the same partition identifier (lines 8–10). After the loop is finished, all ready rules in Π mapped to the same partition identifier are put together, yielding a set of nonempty subprograms $\{\Pi_i\}_{1 \leq i \leq n}$ (lines 12–13).

It can be seen that the number of iterations (lines 3–11) is at most $|\Pi|$, because the mapping adjustment (lines 9–10) ensures that in each iteration, a ready rule r_m having the smallest value of $\min(map(r))$ among $\{r \in \Pi \mid r$ is ready and $|map(r)| > 1\}$ must reach a state that $|map(r_m)| = 1$ and that $map(r_m)$ is unchanged in subsequent iterations. Furthermore, $\Pi_0 = \Pi \setminus \bigcup_{i=1}^{n} \Pi_i$ is the intended set of filtered rules (see Lemma 1); Π_i and Π_j contain no common ground atoms for all $1 \le i < j \le n$. Since Π is sequentially accessed in each iteration, the algorithm is also disk-based.

Lemma 1. *Let $\mathcal{P}$ be the set of subprograms returned by* ***Partition***(*Π, X*) *and $\Pi_0 = \Pi \setminus \bigcup \mathcal{P}$. For any X-MC model M of $\bigcup \mathcal{P}$, $M' = M \cup \bigcup_{r \in \Pi_0} \{\alpha \in head(r) \mid \alpha \notin X, map(\alpha) = 0\}$ is an X-MC model of Π such that $M \cap X = M' \cap X$.*

Proof. Let $M_0 = \bigcup_{r \in \Pi_0} \{\alpha \in head(r) \mid \alpha \notin X, map(\alpha) = 0\}$. Since $M_0 \cap head(r) \neq \emptyset$ for every $r \in \Pi_0$, $M' = M \cup M_0$ satisfies every rule in Π_0. Moreover, since $map(\alpha) > 0$ for every ground atom α occurring in $\bigcup \mathcal{P}$, $M_0 \cap M = \emptyset$ and thus M' still satisfies every rule in $\bigcup \mathcal{P}$ as M. It follows that M' is a model of Π. Since $M_0 \cap X = \emptyset$, M' is an X-MC model of Π such that $M \cap X = M' \cap X$. $\square$

Example 4. Continue with Example 3. Let $X^\sharp$ be the set of ground atoms of the form α^- in $\Pi(KB)$. We now demonstrate how $\texttt{Partition}(\Pi(KB), X^\sharp)$ works. We first move $r_{11}, \ldots, r_{14}, r_{20}, r_{21}$ in front of other rules in $\Pi(KB)$, then map each ground atom in $\Pi(KB)$ to a partition identifier. In the first iteration for processing rules $r \in \Pi(KB)$, $map(r)$ is set as follows (for every ground atom α, $\alpha_{j:k}$ denotes $map(\alpha) = j$ before processing r at line 8 in Algorithm 2, and $map(\alpha) = k$ after the first iteration).

$r_{11} :\leftarrow A(s_2)_{1:1}, me(s_2, s_{2f})_{2:1}, E(s_{2f})_{3:1}$. $\quad r_{12} :\leftarrow T(t_1)_{4:4}$.
$r_{13} :\leftarrow A(s_2)_{1:1}, S(s_2)_{5:1}$. $\quad r_{14} :\leftarrow (t_1 \not\approx t_1)_{6:6}$.
$r_{20} :\leftarrow E(p_1)_{7:7}$. $\quad r_{21} :\leftarrow T(p_2)_{8:7}, E(p_2)_{9:7}$.
$r_1 : S(s_2)_{1:1} \vee S(s_2)^-_{10:1}$. $\quad r_2 : A(s_2)_{1:1} \vee A(s_2)^-_{11:1}$.
$r_3 : (t_1 \not\approx t_1)_{6:6} \vee (t_1 \not\approx t_1)^-_{12:6}$. $\quad r_4 : T(t_1)_{4:4} \vee T(t_1)^-_{13:4}$.
$r_5 : (p_1 \approx p_2)_{14:7} \vee (p_1 \approx p_2)^-_{15:7}$. $r_6 : E(p_2)_{8:7} \vee E(p_2)^-_{16:7}$.
$r_7 : P(s_{2f})_{0:1} \leftarrow S(s_2)_{1:1}$. $\quad r_8 : me(s_2, s_{2f})_{1:1} \leftarrow S(s_2)_{1:1}$.
$r_9 : H(s_2)_{0:0} \leftarrow S(s_2)_{1:1}$. $\quad r_{10} : E(s_{2f})_{1:1} \leftarrow P(s_{2f})_{17:1}$.
$r_{15} : T(p_2)_{8:7} \leftarrow (p_1 \approx p_2)_{14:7}$.
$r_{16} : E(p_2)_{8:7} \leftarrow (p_1 \approx p_2)_{8:7}, E(p_1)_{7:7}$.
$r_{17} : pa(s_1, p_2)_{0:0} \leftarrow (p_1 \approx p_2)_{7:7}$.
$r_{18} : E(p_1)_{7:7} \leftarrow (p_1 \approx p_2)_{7:7}, E(p_2)_{7:7}$.
$r_{19} : pa(s_2, p_2)_{0:0} \leftarrow (p_1 \approx p_2)_{7:7}$.

In the second iteration, it is detected that $|map(r)| = 1$ for all ready rules $r \in \Pi(KB)$, so the loop is finished. Finally we obtain four disjoint subprograms from the resulting mapping: $\Pi_1 = \{r_{11}, r_{13}, r_1, r_2, r_7, r_8, r_{10}\}$, $\Pi_2 = \{r_{12}, r_4\}$, $\Pi_3 = \{r_{14}, r_3\}$ and $\Pi_4 = \{r_{20}, r_{21}, r_5, r_6, r_{15}, r_{16}, r_{18}\}$. $\square$

In what follows, we call a ground atom of the form α^- a *translated decision atom*. Let $\mathcal{C}$ be the set of congruence classes returned by $\texttt{Ground}(\mathcal{R}(KB))$, and $\{\Pi_i\}_{1 \le i \le n}$ the set of subprograms returned by $\texttt{Partition}(\Pi(KB), X^\sharp)$, where $X^\sharp$ is the set of translated decision atoms occurring in $\Pi(KB)$. We intend to compute X-MC models of $\mathcal{R}(KB)$ over each of $\{\Pi_i\}_{1 \le i \le n}$, where X is the set of decision atoms occurring in $\mathcal{R}(KB)$. However, some decision atoms are replaced with other atoms in $\bigcup_{i=1}^{n} \Pi_i$, because all constants in a congruence class in $\mathcal{C}$ are replaced with a representative constant. Let X' be the set of translated decision atoms occurring in $\bigcup_{i=1}^{n} \Pi_i$. X' is said to be *soundly converted* from X w.r.t. $\mathcal{C}$ if each $T(a_1, \ldots, a_k)^- \in X'$ has been given a cost $c'(T(a_1, \ldots, a_k)^-) = \sum_{T(b_1,\ldots,b_k)^- \in X, b_1 \doteq_{\mathcal{C}} a_1, \ldots, b_k \doteq_{\mathcal{C}} a_k} c(T(b_1, \ldots, b_k)^-)$, where $b_i \doteq_{\mathcal{C}} a_i$ means that constants b_i and a_i are the same or together in some $C \in \mathcal{C}$. Such conversion is reasonable because all decision atoms $T(b_1, \ldots, b_k)^- \in X$ such that $b_1 \doteq_{\mathcal{C}} a_1, \ldots, b_k \doteq_{\mathcal{C}} a_k$ for some $T(a_1, \ldots, a_k)^- \in X'$ must be present or absent together in every model of $\mathcal{R}(KB)$. Moreover, given a subset S of X', we define a *decoding* of S w.r.t. X and $\mathcal{C}$, written $d(S, X, \mathcal{C})$, as $\{T(b_1, \ldots, b_k)^- \in X \mid T(a_1, \ldots, a_k)^- \in S, b_1 \doteq_{\mathcal{C}} a_1, \ldots, b_k \doteq_{\mathcal{C}} a_k\}$. Then, a minimum cost diagnosis of KB corresponds to a disjoint union of models in each subprogram (see Theorem 4).

Example 5. Continue with Example 4. Let X be the set of decision atoms in $\mathcal{R}(KB)$. In $\bigcup_{i=1}^{4} \Pi_i$, the set of ground atoms soundly converted from X w.r.t. $\mathcal{C} = \{\{t_1, t_2, s_{1f}\}\}$ is $X' = \{S(s_2)^-, A(s_2)^-, (t_1 \not\approx t_1)^-, T(t_1)^-, (p_1 \approx p_2)^-, E(p_2)^-\}$, where each ground atom $\alpha^- \in X'$ is given a cost $c'(\alpha^-) = 1$ except that $c'(T(t_1)^-) = 2$. Let X_i $(i = 1, \ldots, 4)$ be the subsets of X' that occur in Π_i. It is easy to see that Π_1 has two X_1-MC models $M_{1,1} = \{S(s_2)^-, A(s_2)\}$ and $M_{1,2} = \{S(s_2), A(s_2)^-, P(s_{2f}), me(s_2, s_{2f}), E(s_{2f})\}$; Π_2 has a unique X_2-MC model $M_2 = \{T(t_1)^-\}$; Π_3 has a unique X_3-MC model $M_3 = \{(t_1 \not\approx t_1)^-\}$; Π_4 has two X_4-MC models $M_{4,1} = \{(p_1 \approx p_2)^-, E(p_2)\}$ and $M_{4,2} = \{E(p_2)^-, p_1 \approx p_2, T(p_2)\}$. Hence, $d(M_{1,1} \cap X_1, X, \mathcal{C}) = \{S(s_2)^-\}$; $d(M_{1,2} \cap X_1, X, \mathcal{C}) = \{A(s_2)^-\}$; $d(M_2 \cap X_2, X, \mathcal{C}) = \{T(t_1)^- T(t_2)^-\}$; $d(M_3 \cap X_3, X, \mathcal{C}) = \{(t_1 \not\approx t_2)^-\}$; $d(M_{4,1} \cap X_4, X, \mathcal{C}) = \{(p_1 \approx p_2)^-\}$; $d(M_{4,2} \cap X_4, X, \mathcal{C}) = \{E(p_2)^-\}$. By Theorem 4, we obtain four minimum cost diagnoses for KB w.r.t. R_{KB}: $\{S(s_2), T(t_1), T(t_2), t_1 \not\approx t_2, p_1 \approx p_2\}$, $\{A(s_2), T(t_1), T(t_2), t_1 \not\approx t_2, p_1 \approx p_2\}$, $\{S(s_2), T(t_1), T(t_2), t_1 \not\approx t_2, E(p_2)\}$ and $\{A(s_2), T(t_1), T(t_2), t_1 \not\approx t_2, E(p_2)\}$. $\square$

Theorem 4. *For KB a $\mathcal{SHIQ}$ ontology and $R_{KB} \subseteq KB_{\mathcal{A}}$ a set of removable assertions such that each assertion $\alpha \in R_{KB}$ is given a removal cost $c(\alpha) > 0$, suppose* ***Ground***(*$\mathcal{R}(KB)$*) *returns $(\mathcal{C}, \Pi(KB))$ and* ***Partition***(*$\Pi(KB)$, $X^\sharp$*) *returns $\{\Pi_i\}_{1 \le i \le n}$, where $X^\sharp$ is the set of translated decision atoms occurring in $\Pi(KB)$. Let X' be the set of translated decision atoms occurring in $\bigcup_{i=1}^{n} \Pi_i$ which is soundly converted from $X = \{\alpha^- \mid \alpha \in R_{KB}\}$, and $X_1, \ldots, X_n$ be the subsets of X' that occur in $\Pi_1, \ldots, \Pi_n$ respectively.*

(Soundness) For each X_i-MC model M_i of Π_i $(i = 1, \ldots, n)$, $\{\alpha \mid \alpha^- \in \bigcup_{i=1}^{n} d(M_i \cap X_i, X, \mathcal{C})\}$ is a minimum cost diagnosis for KB w.r.t. R_{KB};

(Completeness) For each minimum cost diagnosis R for KB w.r.t. R_{KB}, there exists an X_i-MC model M_i of Π_i $(i = 1, \ldots, n)$ such that $R = \{\alpha \mid \alpha^- \in \bigcup_{i=1}^{n} d(M_i \cap X_i, X, \mathcal{C})\}$.

Proof sketch. (Soundness) For $i = 1, \ldots, n$, let B_i be the set of ground atoms occurring in Π_i and M_i an X_i-MC model of Π_i. Let $\Pi_0 = \Pi(KB) \setminus \bigcup_{i=1}^{n} \Pi_i$ and $M = \bigcup_{r \in \Pi_0} \{\alpha \in head(r) \mid \alpha \notin X^\sharp \cup \bigcup_{i=1}^{n} B_i\} \cup \bigcup_{i=1}^{n} (M_i \cap B_i)$. Let M' be the set of ground atoms derived by applying all rules in $\mathcal{R}(KB)_\approx$ over $M \cup \{a \approx b \mid a$ and b are together in some $C \in \mathcal{C}\}$, and $M^+ = M \cup M'$. It can be seen that $\bigcup_{i=1}^{n} d(M_i \cap X_i, X, \mathcal{C}) = M^+ \cap X$. It can further be shown that M^+ is an X-MC model of $\mathcal{R}(KB)$. By Theorem 3, $\{\alpha | \alpha^- \in \bigcup_{i=1}^{n} d(M_i \cap X_i, X, \mathcal{C})\} = \{\alpha | \alpha^- \in M^+\}$ is a minimum cost diagnosis for KB w.r.t. R_{KB}.

(Completeness) Let R be a minimum cost diagnosis for KB w.r.t. R_{KB}. By Theorem 3, there exists an X-MC model M of $\mathcal{R}(KB)$ s.t. $R = \{\alpha | \alpha^- \in M\}$. Let M' be a set of ground atoms converted from M by rewriting each constant a in M s.t. $f_c(a, \mathcal{C})$ exists to $min_c(f_c(a, \mathcal{C}))$. It can be shown that $M_i = M' \cap B_i$ is an X_i-MC model of Π_i $(i = 1, \ldots, n)$ s.t. $R = \{\alpha \mid \alpha^- \in \bigcup_{i=1}^{n} d(M_i \cap X_i, X, \mathcal{C})\}$. $\square$

By Theorem 4, the problem of finding minimum cost diagnoses for KB w.r.t. R_{KB} is reduced to n subproblems, each of which computes X_i-MC models of Π_i $(i = 1, \ldots, n)$. We consider computing X_i-MC models of Π_i by applying SAT solvers that support PB-constraints. We assume that the cost of each atom in X_i has been scaled to a positive integer polynomial in $|X|$ the total number of removable assertions. Then, the first X_i-MC model of Π_i can be computed by a binary search (within range $[0, \sum_{\beta \in X_i} c'(\beta)]$) for the minimum value v_{min} such that $\Pi_i \cup \{\sum_{\beta \in X_i} c'(\beta) \cdot \text{assign}(\beta) \leq v_{min}\}$ is satisfiable, taking $O(\log_2 \sum_{\beta \in X_i} c'(\beta)) = O(\log_2 |X|)$ calls to a SAT solver. Let $\mathcal{M} = \{M \cap X_i \mid M$ is a previously computed X_i-MC model of $\Pi_i\}$. Then a next X_i-MC model M of Π_i, such that $M \cap X_i \neq S$ for every $S \in \mathcal{M}$, can be computed as a model of $\Pi_i \cup \{\sum_{\beta \in X_i} c'(\beta) \cdot \text{assign}(\beta) \leq v_{min}\} \cup \{\leftarrow \bigwedge_{\beta \in S} \beta \mid S \in \mathcal{M}\}$, by calling a SAT solver once.

Consider the time complexity for computing minimum cost diagnoses. Under the data complexity assumption, the number of non-ground rules in $\mathcal{R}(KB)$ and the number of different variables in each rule in $\mathcal{R}(KB)$ are bounded by constants. Thus the number of rules in $\Pi(KB)$ is polynomial in $|KB_{\mathcal{A}}|$. It follows that `Ground`$(\mathcal{R}(KB))$ is executed in PTIME. Let $X^\sharp$ be the set of translated decision atoms occurring in $\Pi(KB)$. `Partition`$(\Pi(KB),\ X^\sharp)$ commits at most $|\Pi(KB)|$ iterations over $\Pi(KB)$, so it is executed in PTIME too. Let n be the number of removable assertions in KB and $\{\Pi_i\}_{1 \leq i \leq m}$ be the set of propositional programs returned by `Partition`$(\Pi(KB),\ X^\sharp)$. Note that the SAT problem with a PB-constraint is NP-complete. Since Π_i and Π_j have no common ground atoms for all $1 \leq i < j \leq m$, m calls to a SAT solver over $\Pi_1, \ldots, \Pi_m$ respectively amount to one call to an NP oracle over $\bigcup_{i=1}^{m} \Pi_i$. Under the assumption that each removal cost has been scaled to a positive integer polynomial in n, it follows from Theorem 4 that, the first minimum cost diagnosis is computed in $O(\log_2 n)$ calls to an NP oracle, and a next one, in one more call.

6. EXPERIMENTAL EVALUATION

We implemented the proposed method for computing minimum cost diagnoses in GNU C++. In the implementation, MySQL is used as the back-end SQL engine; ABox assertions and new ground atoms derived in the grounding process are maintained in a SQL database; All ground substitutes of rules, retrieved via SQL statements, are maintained in disk files; The SAT solver MiniSat$^+$ [5], which supports PB-constraints and linear optimization functions by internally translating them into SAT clauses, is applied to compute X-MC models. All the experiments were conducted on a 3.2GHz Pentium 4 CPU 2GB RAM PC running Windows XP and Cygwin.

6.1 Test Ontologies and Preparations

Semintec[2] is an ontology about financial services, created in the SEMINTEC project at the University of Poznan. Its intensional part contains functional roles and disjointness constraints.

[2]http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm

Table 1: The complexity of test ontologies

	N_C	N_R	N_I	N_A	N_r	Features
S	59	16	17,941	65,291	221	EQ$_{13}$,DS$_0$
H	27	49	82,095	154,110	159	EQ$_7$,DS$_7$
L1	86	34	50,253	120,274	168	EQ$_2$,DS$_0$
L10	86	34	629,568	1,293,286	168	EQ$_2$,DS$_0$

Note: S stands for Semintec. H stands for HumanCyc$^-$. L1 stands for LUBM1$^+$. L10 stands for LUBM10$^+$. N_C is the number of concept names. N_R is the number of role names. N_I is the number of individuals. N_A is the number of ABox assertions stored in MySQL databases. N_r is the number of rules transformed from the intensional part. The features show how many special transformed rules: EQ$_n$ means there are n *equality rules* (i.e. rules containing equality atoms); DS$_n$ means there are n disjunctive rules.

HumanCyc[3] is an ontology on human metabolic pathways and human genome, created by the SRI International corporation. Since its intensional part contains nominals and concrete domain specifications (e.g., a role range is of some datatype, a concrete role is functional, etc.) that cannot be handled by our method, we converted nominals to new atomic concepts and deleted concrete domain specifications. The weakened intensional part still contains disjunctions, functional roles/restrictions and disjointness constraints.

LUBM[4] is a benchmark ontology developed at the Lehigh University [9]. Since its intensional part has no functional roles, number restrictions or disjointness constraints, which implies that it cannot be inconsistent, we extended it by adding a functional role `headOf` and an inverse functional role `isTaughtBy`, where `headOf` (resp. `isTaughtBy`) is also defined as an inverse role of `isHeadOf` (resp. `teacherOf`), and adding disjointness constraints $X \sqsubseteq \neg NonX$ for each existing concept name X, where $NonX$ is a new concept name. LUBM comes with an ABox generator. Let LUBMn denote the ontology obtained from the generator by setting the number of universities to n.

Before testing the proposed method, the intensional parts of the above ontologies were offline transformed to datalog programs using the KAON2 DL reasoner[5]. Each transformation was performed in less than one second. Moreover, ABox assertions of the above ontologies were stored into MySQL databases, where duplicated ABox assertions were removed. Table 1 summarizes the complexity of the test ontologies and the datalog programs transformed from their intensional parts, where HumanCyc$^-$ denotes the weakened HumanCyc, and LUBMn^+ denotes the extended LUBMn.

We developed a tool, called `Injector`, to inject a given number of conflicts into an ontology. Given a consistent ontology KB and a number n_{cnf} of conflicts to be injected, `Injector` first deduces into KB all atomic concept assertions that are consequences of KB, then injects n_{cnf} conflicts one by one. Let S_{FR} denote the set of functional/inverse func-

[3]http://humancyc.org/

[4]http://swat.cse.lehigh.edu/projects/lubm/

[5]http://kaon2.semanticweb.org/

tional roles, S_{DC} denote the set of atomic concepts that have disjoint concepts. To inject a conflict, `Injector` randomly selects an entity in $S_{FR} \cup S_{DC}$. In case an functional role R is selected, if there exist role assertions over R in KB, `Injector` randomly selects one, say $R(a, b)$, and appends $R(a, c)$ and $b \not\approx c$ to KB, where c is a new individual; otherwise, `Injector` appends $R(a, b)$, $R(a, c)$ and $b \not\approx c$ to KB, where a, b, c are new individuals. In case an inverse functional role R is selected, `Injector` treats it as R^-. In case an atomic concept C is selected, if there exist concept assertions over C in KB, `Injector` randomly selects one, say $C(a)$, and appends $D(a)$ to KB for a randomly selected disjoint concept D of C; otherwise, `Injector` appends $C(a)$ and $D(a)$ to KB, where a is a new individual and D a randomly selected disjoint concept of C. `Injector` was implemented in JAVA, using the Pellet [29] API to deduce all atomic concept assertions that are consequences of a consistent ontology.

6.2 Experimental Results

We injected different number of conflicts to the four test ontologies using `Injector`, obtaining a set of inconsistent ontologies. We consider the hardest case where all assertions in an obtained ontology are assumed removable. We assume that each assertion is given a removal cost 1. In order to make the implemented system terminate in an acceptable time, we set a time limit of 20 minutes for one call to MiniSat$^+$.

The test results are reported in Table 2. In each block, the first row lists n_{cnf}, i.e. the number of injected conflicts; the second row lists the total execution time for computing the first minimum cost diagnosis, starting from transforming the input ontology. For Semintec and HumanCyc$^-$, the implemented system cannot handle more injected conflicts in our test environment, because when $n_{cnf} = 140$ for Semintec (or $n_{cnf} = 60$ for HumanCyc$^-$), some call to MiniSat$^+$ exceeds the time limit. In contrast, the implemented system scales well on LUBM1$^+$/LUBM10$^+$ ontologies with increasing number (from 1000) of conflicts.

We also collected runtime statistics on the partitioning process. Let KB be an inconsistent ontology reported in Table 2. As can be seen, the number of rules in each grounded repair program $\Pi(KB)$ is up to millions. In addition, the number of decision atoms in $\Pi(KB)$ is at least in thousands. We experimentally verified that any $\Pi(KB)$, without being partitioned, cannot be handled by MiniSat$^+$ because the execution exceeds the time or memory limit. This shows the importance of the partitioning process.

Other statistics show the effectiveness of the partitioning process. The percentage of filtered rules, $\frac{|\Pi_0|}{|\Pi(KB)|}$, is at least 11% for all reported ontologies (esp., at least 57% for LUBM10$^+$ ontologies). The number of disjoint subprograms of $\Pi(KB)$ (i.e. the number of partitions), $\#\{\Pi_i\}$, increases when the number of conflicts increases. This shows that the partitioning process improves the scalability.

Table 2 also reports the maximum number of ground rules in each partition, $\max(|\{\Pi_i\}|)$, and the maximum number of translated decision atoms in each partition, $\max(|\{X_i\}|)$. It can be seen that the total execution time is roughly dominated by $\max(|\{\Pi_i\}|)$ and $\max(|\{X_i\}|)$. For Semintec and HumanCyc$^-$, since $\max(|\{X_i\}|)$ is up to tens of thousands, the execution of MiniSat$^+$ over the largest partition quickly exceeds the time limit when the number of conflicts increases (as $\max(|\{\Pi_i\}|)$ increases too). In contrast, for LUBM1$^+$ and LUBM10$^+$, since $\max(|\{\Pi_i\}|)$ or $\max(|\{X_i\}|)$ is stable around a relatively small value, the total execution time increases smoothly when the number of conflicts increases.

We can conclude that the performance of our method depends on the effectiveness of the partitioning process. As for what influences such effectiveness when the number of assertions and the number of conflicts are fixed, we can see from Table 1 and Table 2 that, equality rules have a stronger impact than normal rules; further, disjunctive rules have a stronger impact than equality rules. Hence, we believe that our method can handle any real (populated) ontology that has up to millions of assertions together with a moderately complex intensional part, which can be transformed to up to hundreds of datalog rules with a few disjunctive rules and equality rules.

Table 2: Test results against different number of conflicts n_{cnf}

Semintec					
n_{cnf}	40	60	80	100	120
Time (sec)	53	191	155	298	1144
$\|\Pi(KB)\|$	159K	241K	230K	371K	636K
$\frac{\|\Pi_0\|}{\|\Pi(KB)\|}$	73.8%	45.1%	53.1%	35.8%	24.9%
$\#\{\Pi_i\}$	25	31	40	49	57
$\max(\{\|\Pi_i\|\})$	11K	66K	76K	152K	393K
$\max(\{\|X_i\|\})$	7054	16114	14687	20241	18794
HumanCyc$^-$					
n_{cnf}	10	20	30	40	50
Time (sec)	428	326	531	412	867
$\|\Pi(KB)\|$	598K	607K	620K	604K	639K
$\frac{\|\Pi_0\|}{\|\Pi(KB)\|}$	26.0%	25.6%	25.1%	25.7%	24.4%
$\#\{\Pi_i\}$	5	7	10	19	21
$\max(\{\|\Pi_i\|\})$	443K	451K	465K	449K	483K
$\max(\{\|X_i\|\})$	68155	68159	68175	68181	68197
LUBM1$^+$					
n_{cnf}	1000	2000	3000	4000	5000
Time (sec)	202	387	554	814	1010
$\|\Pi(KB)\|$	537K	978K	1490K	2306K	2877K
$\frac{\|\Pi_0\|}{\|\Pi(KB)\|}$	41.9%	25.9%	18.7%	13.3%	11.4%
$\#\{\Pi_i\}$	886	1766	2696	3645	4606
$\max(\{\|\Pi_i\|\})$	43K	146K	120K	244K	540K
$\max(\{\|X_i\|\})$	808	859	895	901	898
LUBM10$^+$					
n_{cnf}	1000	2000	3000	4000	5000
Time (sec)	736	851	1070	1250	1618
$\|\Pi(KB)\|$	3893K	4111K	4165K	4338K	4753K
$\frac{\|\Pi_0\|}{\|\Pi(KB)\|}$	82.3%	70.8%	65.8%	62.8%	57.3%
$\#\{\Pi_i\}$	963	1860	2713	3605	4423
$\max(\{\|\Pi_i\|\})$	26K	24K	31K	43K	34K
$\max(\{\|X_i\|\})$	810	1571	1961	2610	1393

Note: $|\Pi(KB)|$ is the number of rules in the grounded repair program $\Pi(KB)$. $\frac{|\Pi_0|}{|\Pi(KB)|}$ is the percentage of rules filtered out in our partitioning algorithm. $\#\{\Pi_i\}$ is the number of partitions. $\max(\{|\Pi_i|\})$ is the maximum number of ground rules in each partition. $\max(\{|X_i|\})$ is the maximum number of translated decision atoms in each partition.

7. CONCLUSION AND FUTURE WORK

A DL-based ontology may become inconsistent after it is populated. In this paper, we proposed a solution to repair the populated ontology by deleting assertions in a minimum cost diagnosis. We first showed the intractability of finding a minimum cost diagnosis, then presented an exact method for computing minimum cost diagnoses for $\mathcal{SHIQ}$ ontologies. The method transforms a $\mathcal{SHIQ}$ ontology to a set of disjoint propositional programs in PTIME w.r.t. data complexity, thus reducing the original problem into a set of independent subproblems. Each such subproblem computes an X-MC model and is solvable by applying SAT solvers. We experimentally showed that the method can handle moderately complex ontologies with over thousands of assertions, where all assertions can be assumed removable. Especially, the method scales well on the extended LUBM ontologies with increasing number (from 1000) of conflicts.

For future work, we plan to enhance our method to cope with concrete domain specifications, seek feasible approaches to handling nominals, and work on tractable approximate methods for computing minimum cost diagnoses.

8. ACKNOWLEDGEMENTS

This work is supported in part by NSFC grants 60673103, 60721061 and 60373052.

9. REFERENCES

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Dl-lite: Tractable description logics for ontologies. In *Proc. of AAAI'05*, pages 602–607, 2005.

[3] P. Cimiano and J. Völker. Text2onto - a framework for ontology learning and data-driven change discovery. In *Proc. of NLDB'05*, pages 227–238, 2005.

[4] J. Dolby, J. Fan, A. Fokoue, A. Kalyanpur, A. Kershenbaum, L. Ma, J. W. Murdock, K. Srinivas, and C. A. Welty. Scalable cleanup of information extraction data using ontologies. In *Proc. of ISWC'07*, pages 100–113, 2007.

[5] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into sat. *JSAT*, 2:1–26, 2006.

[6] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Trans. Database Systems*, 22(3):364–418, 1997.

[7] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A deductive system for non-monotonic reasoning. In *Proc. of LPNMR'97*, pages 364–375, 1997.

[8] M. Fitting. *First-order Logic and Automated Theorem Proving (2nd ed.)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[9] Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2–3):158–182, 2005.

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3), 2000.

[11] Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI'05*, pages 454–459, 2005.

[12] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In *Proc. of KR'04*, pages 152–162, 2004.

[13] A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in owl ontologies. In *Proc. of ESWC'06*, pages 170–184, 2006.

[14] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in owl ontologies. *J. Web Sem.*, 3(4):268–293, 2005.

[15] R. M. Karp. Reducibility among combinatorial problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[16] S. C. Lam, J. Z. Pan, D. H. Sleeman, and W. W. Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *Proc. of WI'06*, pages 428–434, 2006.

[17] D. Lembo and M. Ruzzi. Consistent query answering over description logic ontologies. In *Proc. of RR'07*, pages 194–208, 2007.

[18] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[19] T. Meyer, K. Lee, and R. Booth. Knowledge integration for description logics. In *Proc. of AAAI'05*, pages 645–650, 2005.

[20] T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of AAAI'06*, pages 269–274, 2006.

[21] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität karlsruhe, Germany, Jan. 2006.

[22] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation, 2004. `http://www.w3.org/TR/owl-semantics/`.

[23] G. Qi, W. Liu, and D. A. Bell. Knowledge base revision in description logics. In *Proc. of JELIA'06*, pages 386–398, 2006.

[24] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[25] S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proc. of ESWC'05*, pages 226–240, 2005.

[26] S. Schlobach. Diagnosing terminologies. In *Proc. of AAAI'05*, pages 670–675, 2005.

[27] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI'03*, pages 355–362, 2003.

[28] H. M. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-boolean sat solver. *JSAT*, 2:157–181, 2006.

[29] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

[30] C. Welty and J. W. Murdock. Towards knowledge acquisition from information extraction. In *Proc. of ISWC'06*, pages 152–162, 2006.