A General Diagnosis Method for Ontologies

Gerhard Friedrich and Kostyantyn Shchekotykhin

Universitaet Klagenfurt
Universitaetsstrasse 65
9020 Klagenfurt, Austria, Europe
firstname.lastname@ifit.uni-klu.ac.at

Abstract. The effective debugging of ontologies is an important prerequisite for their successful application and impact on the semantic web. The heart of this debugging process is the diagnosis of faulty knowledge bases. In this paper we define general concepts for the diagnosis of ontologies. Based on these concepts, we provide correct and complete algorithms for the computation of minimal diagnoses of knowledge bases. These concepts and algorithms are broadly applicable since they are independent of a particular variant of an underlying logic (with monotonic semantics) and independent of a particular reasoning system. The practical feasibility of our method is shown by extensive test evaluations.

1 Introduction

Ontologies are playing a key role for the successful implementation of the Semantic Web. Various languages for the specification of ontologies were proposed. The W3C Web Ontology working group has developed OWL [1] which is currently the language of choice for expressing Semantic Web ontologies. In fact OWL consists of three languages of increasing expressive power: OWL Lite, OWL DL and OWL Full. For the two decidable languages OWL Lite and OWL DL the strong relation to description logics was shown in [2]. OWL Lite and OWL DL are basically very expressive description logics built upon RDF Schema. Based on these methods efficient reasoning services for OWL Lite can be provided by systems like RACER [3].

Hand in hand with the increase of applications of ontologies and their growing size, the support of ontology development becomes an important issue for a broad and successful technology adoption. In the development phase of ontologies, testing and debugging is a major activity. Testing validates if the actual knowledge base matches the intended meaning of the knowledge engineer. In case of errors, the knowledge engineer has to debug the knowledge base. In this debugging process, the knowledge base must be diagnosed and changed such that all test cases are successfully passed. Consequently, the diagnosis process has to identify sets of axioms (preferable minimal sets) which should be changed in order to match the requirements expressed in tests.

In order to support the debugging process current work focuses on the identification of sets of axioms which are responsible for an incoherent (rsp. inconsistent) knowledge base [4, 5]. We enhance current techniques in several lines.

First, we will provide a general definition of the diagnosis problem employing a broadly accepted theory of diagnosis. On the bases of this theory we introduce test

cases which allow the knowledge engineer to formulate application specific requirements. Furthermore, this general theory of diagnosis allows the diagnosis of incoherent and inconsistent knowledge bases which comprise both terminological and assertional axioms. Second, we will show that concepts introduced in [4] are special cases of the proposed diagnosis theory. In addition, we argue that the concept of *minimal diagnoses* should be preferred over *cores*, if the goal is to find minimal changes of the knowledge base. Third, we provide correct and complete algorithms for the computation of minimal diagnoses. These algorithms are independent of a particular variant of a logic with monotonic semantics and work with arbitrary reasoning systems. Forth, we evaluate our algorithms employing standard test libraries showing the feasibility of our methods.

The reminder of the paper is organized as follows: In order to make the paper self contained Section 2 provides a brief introduction to the main concepts of description logic. Section 3 presents an introductory example for the diagnosis of ontologies. The basic concepts and properties of our approach are introduced in Section 4. Section 5 describes the algorithms for the computation of minimal diagnoses and minimal conflicts, followed by the presentation of our evaluation results in Section 6. The paper closes with a discussion of related work.

2 Description Logics

Since the underlying knowledge representation method of ontologies in the Semantic Web is based on description logics we introduce briefly the main concepts. For our investigation we employ the usual definition of description logics as defined in [6,7]. A knowledge base comprises two components a TBox (terminology $\mathcal{T}$) and a ABox ($\mathcal{A}$). The TBox defines the terminology whereas the Abox contains assertions about named individuals in terms of a vocabulary defined in the TBox. The vocabulary consists of concepts, denoting sets of individuals, and roles, denoting binary relationships between individuals. These concepts and roles may be either atomic or complex. Complex descriptions are obtained by employing description operators. The language of descriptions is defined recursively by starting from a schema $S = (\mathcal{CN}, \mathcal{RN}, \mathcal{IN})$ of disjoint sets of names for concepts, roles, and individuals. Typical operators for the construction of complex descriptions are $C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), $\neg C$ (negation), $\forall R.C$ (concept value restriction), and $\exists R.C$ (concept exists restriction), where C and D are concept descriptions and $R \in \mathcal{RN}$.

Knowledge bases are defined by a finite set of assertions. Assertions regarding the TBox are called terminological axioms. Assertions regarding the ABox are called assertional axioms. Terminological axioms are expressed by $C \sqsubseteq D$ (Generalized Concept Inclusion) which corresponds to the logical implication. Let $a, b \in \mathcal{IN}$ be individual names then $C(a)$ and $R(a, b)$ are assertional axioms.

Concepts (rsp. roles) can be regarded as unary (rsp. binary) predicates. Roughly speaking description logics can be seen as fragments of first-order predicate logic (without considering transitive closure or special fixpoint semantics). These fragments are specifically designed to assure decidability or favorable computational costs.

The semantics of description terms are usually given denotationally using an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, (\cdot)^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a domain (non-empty universe) of values, and

$(\cdot)^{\mathcal{I}}$ a mapping from concept descriptions to subsets of the domain, and from role descriptions to sets of 2-tuples over the domain. The mapping also associates with every individual name in $\mathcal{IN}$ some value in $\Delta^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is a model of a knowledge base iff it satisfies all terminological axioms and assertional axioms. A knowledge base is satisfiable iff there exists a model.

A description E is coherent w.r.t. a TBox $\mathcal{T}$, if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $E^{\mathcal{I}} \neq \emptyset$. A TBox is incoherent iff there exists an incoherent concept or role.

3 Example

For the debugging of a knowledge base KB, we distinguish two basic operations. The first operation is the deletion of axioms and the second operation deals with the addition of axioms. Changes of axioms can be viewed as combined add/delete operations. Diagnosis deals with the first operation, i.e. the identification of axioms which must be changed (deleted) whereas repair deals with the addition of appropriate axioms.

Knowledge bases are designed in order to provide reasoning services. In classical logical systems such reasoning services assume a satisfiable knowledge base. Consequently, restoring consistency of unsatisfiable knowledge bases is a major goal in debugging. In addition, the coherence of knowledge bases may be required.

Furthermore, knowledge bases may be tested by employing test cases. These test cases are formulated by the knowledge engineer and define requirements for the knowledge base. A test case is a set of test axioms. For example, we may exploit assertional axioms to validate a knowledge base. In the configuration domain we may test the knowledge base if a set of requirements (a set of key components) leads to an intended configuration which assures certain properties.

Let us assume we test the following knowledge base KB_E which is a variant of the example provided by [4].

$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3$	$ax_2 : A_2 \sqsubseteq \neg D \sqcap A_4$
$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5$	$ax_4 : A_4 \sqsubseteq \forall s.F \sqcap C$
$ax_5 : A_5 \sqsubseteq \exists s.\neg F$	$ax_6 : A_6 \sqsubseteq A_4 \sqcap D$

In addition, we define a background theory $B_E = \{A_6(w), A_1(u), s(u, v)\}$ which is considered as correct.

In the following we assume that the knowledge engineer formulates requirements (test axioms). The goal of the diagnosis process is to find subsets of axioms which *must* be changed such that all requirements (test cases) can be met. We will characterize these sets of axioms by minimal (irreducible) sets. Of course the knowledge engineer may decide to change supersets of these minimal sets, e.g. because the knowledge base should reflect the mental model of the knowledge engineer as close as possible. However, the incorporation of mental models for generating diagnoses and repairs is out of the scope of this paper. Therefore, we use symbols in our example that have no predefined intended interpretation. The intended interpretation is solely defined by the knowledge base and the test cases.

Let us assume we require a coherent knowledge base (Requirement 1). In our example the knowledge base is incoherent (i.e. A_1 and A_3 are incoherent). The irreducible set

of axioms which preserves the incoherence of the knowledge base is $\langle ax3, ax4, ax5\rangle$ [1] (i.e. A_3 is incoherent). It follows that at least one of these axioms must be changed in order to fulfill Requirement 1.

If we in addition require that $KB_E \cup B_E$ is consistent with the assertional test axiom $\neg C(w)$ (Requirement 2) then an additional irreducible set of axioms of the knowledge base which is unsatisfiable with the test axiom is $\langle ax4, ax6\rangle$. Similar to the previous case, one of these axioms must be changed. In order to achieve satisfiability and coherence with *minimal* changes, we have to change at least either axiom $ax4$ or the axioms $[ax3, ax6]$ or $[ax5, ax6]$.

Let us assume Requirement 3 says that $F(v)$ must be unsatisfiable with $KB_E \cup B_E$. In case where we consider $ax4$ to be faulty, then it is possible to fulfill all the requirements, i.e. we can delete $ax4$ and find an extension of KB_E to satisfy all requirements including Requirement 3. A trivial extension to satisfy Requirement 3 is to add $\neg F(v)$ to KB_E.

However, in the cases where we consider either $[ax3, ax6]$ or $[ax5, ax6]$ to be changed then all 3 requirements could not be satisfied since $KB_E \cup B_E - [ax3, ax6] \models F(v)$ and also $KB_E \cup B_E - [ax5, ax6] \models F(v)$. In order to satisfy Requirement 3 in addition to Requirements 1 and 2 one of the axioms in $\langle ax1, ax2, ax4\rangle$ and in $\langle ax1, ax3, ax4\rangle$ must be changed. Consequently, the minimal change in order to satisfy all requirements is to replace $ax4$ (e.g. by $A_4 \sqsubseteq \forall s.\neg F \sqcap \neg C$). All other *minimal* changes involve at least 3 axioms, e.g. $[ax1, ax3, ax6]$.

In the next section we will develop a general theory for the diagnosis of logic-based ontologies.

4 Diagnosis of ontologies

The goal of the diagnosis process is to identify those axioms which cause faults. Such axioms are considered as the cause of faults iff the knowledge base without these axioms is not faulty. What is regarded as fault depends on properties defined by the knowledge engineer. In knowledge bases which are based on logical descriptions usually satisfiability is a necessary property. In addition the knowledge engineer may specify a test case by a set of axioms. In the following we regard these axioms as correct. Of course the formulation of these test cases is restricted by the expressive power of the underlying language.

Let the set of test cases TST be partitioned in 4 disjoint sets TC^+, TC^-, TI^+, and TI^-. We can distinguish four different scenarios for testing.

1. $KB \cup e^+$ consistent, $\forall e^+ \in TC^+$
2. $KB \cup e^-$ inconsistent, $\forall e^- \in TC^-$
3. $KB \models ne^-, \forall ne^- \in TI^-$
4. $KB \not\models ne^+, \forall ne^+ \in TI^+$

[1] According to the terminology used in model-based diagnosis such a set is called a conflict set. For denoting conflict sets (rsp. diagnoses) we use the notation $\langle \ldots \rangle$ (rsp. $[\ldots]$) employed in model-based diagnosis.

By exploiting negation the third case is equivalent to the second by checking if $KB \cup \neg ne^-$ is unsatisfiable. Likewise, the forth case can be reduced to the first case by checking if $KB \cup \neg ne^+$ is satisfiable. Therefore, (without limiting the generality) we will consider only cases 1 and 2.

Please note that requiring coherence of a knowledge base corresponds to the specification of appropriate test axioms. Formulated in predicate logic this means we require $\{\{\exists X : C(X)\}|C \in \mathcal{CN}\}$ as a set of test axioms contained in TC^+. For presentation purposes we refer to this set of axioms by ax_{co}. For the coherence of roles (e.g. for DLs with role constructors) ax_{ro} is $\{\{\exists X, Y : r(X, Y)\}|r \in \mathcal{RN}\}$.

In the following we will extend the approach of diagnosing configuration knowledge bases presented in [8] to logical knowledge bases. In addition, we will allow the definition of a background theory (represented as a set of axioms) which is considered to be correct. One reason for the introduction of a background theory is, that during the debugging process, the knowledge engineer may define some axioms as correct and therefore these axioms should not be included in any diagnoses.

Definition 1. ***KB-Diagnosis Problem:*** *A KB-Diagnosis Problem (Diagnosis Problem for a Knowledge Base) is a tuple* $(KB, B,\ TC^+,\ TC^-)$ *where* KB *is a knowledge base, B a background theory,* TC^+ *is a set of positive and* TC^- *a set of negative test cases. The test cases are given as sets of logical sentences. We assume that each test case on its own is consistent.*

The principle idea of the following definition is to find a set of axioms of the knowledge base which must be changed (respectively deleted) and, eventually, some axioms must be added such that all test cases are satisfied.

Definition 2. ***KB-Diagnosis****: A KB-Diagnosis for a KB-Diagnosis Problem* $(KB, B,\ TC^+,\ TC^-)$ *is a set* $S \subseteq KB$ *of sentences such that there exists an extension* EX*, where* EX *is a set of logical sentences added to the knowledge base, such that*

1. $(KB - S) \cup B \cup EX \cup e^+$ *consistent* $\forall e^+ \in TC^+$
2. $(KB - S) \cup B \cup EX \cup e^-$ *inconsistent* $\forall e^- \in TC^-$

Note, that an extension may be needed to achieve inconsistency with the test cases of TC^-. If we assume that we are interested in minimal changes of the existing axioms (i.e. it is more likely that an axiom is correct than it is incorrect) then we are especially interested in minimal (irreducible) diagnoses. In addition, these minimal diagnoses are exploited to characterize the set of all diagnoses.

Definition 3. ***Minimal KB-Diagnosis****: A KB-Diagnosis S for a KB-Diagnosis Problem* $(KB, B,\ TC^+,\ TC^-)$ *is minimal iff there is no proper subset* $S' \subset S$ *s.t.* S' *is a diagnosis.*

Definition 4. ***Minimum cardinality KB-Diagnosis****: A KB-Diagnosis S for a KB-Diagnosis Problem* $(KB, B,\ TC^+,\ TC^-)$ *is a minimum cardinality diagnosis iff there is no diagnosis* S' *s.t.* $|S'| < |S|$.

In the following we assume the monotonic semantics of standard logic. A diagnosis will always exist under the (reasonable) condition that background theory, positive test cases, and negative test cases do not interfere with each other. The following proposition allows us to characterize diagnoses without the extension EX. The idea is to use the negative examples to define this extension.

Proposition 1. *Given a KB-Diagnosis Problem* (KB, B, TC^+, TC^-)*, a diagnosis* S *for* (KB, B, TC^+, TC^-) *exists iff* $\forall e^+ \in TC^+ : e^+ \cup B \cup \bigwedge_{e^- \in TC^-}(\neg e^-)$ *is consistent.*

From here on, we refer to the conjunction of all negated negative test cases as NE, i.e. $NE = \bigwedge_{e^- \in TC^-}(\neg e^-)$.

Corollary 1. S *is a diagnosis for* (KB, B, TC^+, TC^-) *iff* $\forall e^+ \in TC^+ : (KB - S) \cup B \cup e^+ \cup NE$ *is consistent.*

Example: Let ${TC^+}_E = \{\{\neg C(w)\} \cup ax_{co}\}$ and ${TC^-}_E = \{\{F(v)\}\}$. The minimal diagnoses of $(KB_E, B_E, {TC^+}_E, {TC^-}_E)$ are $[ax4]$, $[ax1, ax3, ax6]$, $[ax1, ax5, ax6]$, and $[ax2, ax3, ax6]$.

As a consequence, every superset of a minimal diagnosis is a diagnosis. Therefore, the set of all diagnoses is characterized by the set of minimal diagnoses, i.e. at least the elements of a minimal diagnosis must be changed.

In order to compute minimal diagnoses we exploit the concept of conflict sets.

Definition 5. ***Conflict Set:*** *A conflict set* CS *for* (KB, B, TC^+, TC^-) *is a set of elements of the knowledge base* $CS \subseteq KB$ *such that* $\exists e^+ \in TC^+ : CS \cup B \cup e^+ \cup NE$ *is inconsistent.*

Definition 6. ***Minimal Conflict Set****: A conflict set* CS *for* (KB, B, TC^+, TC^-) *is minimal iff there is no proper subset* $CS' \subset CS$ *s.t.* CS' *is a conflict.*

Example: The minimal conflict sets for $({KB}_E, B_E, {TC^+}_E, {TC^-}_E)$ are $\langle ax3, ax4, ax5\rangle$, $\langle ax4, ax6\rangle$, $\langle ax1, ax2, ax4\rangle$, and $\langle ax1, ax3, ax4\rangle$.

The following proposition (which follows from results of [9]) shows the relation between minimal conflict sets and minimal diagnoses. It is based on the observation that at least one element from each minimal conflict must be changed.

Proposition 2. *Provided that there exists a diagnosis for* (KB, B, TC^+, TC^-)*.* S *is a minimal diagnosis for* (KB, B, TC^+, TC^-) *iff* S *is a minimal hitting set for the set of all minimal conflict sets of* (KB, B, TC^+, TC^-)*.*

For the debugging of incoherent TBoxes without test cases and background theory [4] introduces the concept of *minimal incoherence-preserving sub-TBox (MIPS)* which corresponds to the concept of *conflict sets* (see [9]) of model-based diagnosis.

Definition 7. ***Minimal incoherence-preserving sub-TBox [4]****: Let* $\mathcal{T}$ *be an incoherent TBox. A TBox* $\mathcal{T}' \subseteq \mathcal{T}$ *is a minimal incoherence-preserving sub-TBox (MIPS) of* $\mathcal{T}$ *if* $\mathcal{T}'$ *is incoherent, and every sub-TBox* $\mathcal{T}'' \subset \mathcal{T}'$ *is coherent.*

Let $TC^{+}{}_{MIPS} = ax_{co} \cup ax_{ro}$.

Proposition 3. *Let $\mathcal{T}$ be the TBox of a knowledge base KB. M is a MIPS of $\mathcal{T}$ iff M is minimal conflict set of $(\mathcal{T}, \emptyset, TC^{+}{}_{MIPS}, \emptyset)$.*

Based on the concept of MIPS [4] defines the concept of *cores*. Cores are sets of axioms occurring in several of these incoherent TBoxes. The rational is that the more MIPS such a core belongs to, the more likely its axioms will be the cause of contradictions. Similar ideas (however with a different intention) were formulated in [10].

Definition 8. ***MIPS-Core [4]****: Let $\mathcal{T}$ be a TBox. A non-empty intersection of n different MIPS of the MIPS of $\mathcal{T}$ (with $n \geq 1$) is called a MIPS-core of arity n for $\mathcal{T}$.*

Under the assumption that the correctness of axioms is more likely than their faultiness, we are interested in minimal diagnoses with a small cardinality. These minimal diagnoses define minimal sets of axioms to be changed. Unfortunately elements of cores with maximal arity may not be included in such diagnoses.

Remark 1. Let $CORE$ be a core of $\mathcal{T}$ with maximal arity. Let $MINDIAG$ be the set of minimal cardinality diagnoses of $(\mathcal{T}, \emptyset, TC^{+}{}_{MIPS}, \emptyset)$. It could be the case that $CORE$ does not contain any element of any minimal cardinality diagnosis, i.e. $CORE \cap S_i = \emptyset$ for all $S_i \in MINDIAG$.

Example: Consider the minimal conflict sets $C_1{:}\langle a, d\rangle, C_2{:}\langle b, e\rangle, C_3{:}\langle c, f\rangle, C_4{:}\langle a, x\rangle$, $C_5{:}\langle b, x\rangle$, and $C_6{:}\langle c, x\rangle$. The arity of core $\{x\}$ is 3 (maximal). All other cores have a lower arity than 3. However, the set of minimal cardinality diagnoses is $\{[a, b, c]\}$. x is only contained in minimal diagnoses with cardinality 4, e.g. $[x, d, e, f]$.

Consequently, cores may point to axioms which need not be changed. In order to discover a minimal number of axioms which must be changed, we therefore propose the computation of minimal diagnoses. Of course the knowledge engineer might decide to change additional axioms based on her design goals (e.g. readability of the knowledge base).

5 Computing minimal diagnoses

For the computation of minimal diagnoses one of our major design goal is generality of our methods. In particular, our only prerequisite is a reasoning system which correctly outputs *consistent* (rsp. *inconsistent*) if a set of sentences is consistent (rsp. inconsistent). Consequently, we neither employ any restriction regarding the variant of a knowledge representation language (based on the standard monotonic semantics) nor restrictions on the knowledge bases (e.g. acyclic).

The principle idea of our approach is to employ Reiter's Hitting Set (HS) algorithm [9] for the computation of a HS-tree. However, this algorithm has the drawback that it degrades rapidly if the underlying reasoning system does not output minimal conflict sets. In the worst case some minimal diagnoses may be missed as pointed out by [11] who proposed a DAG-variant of the original algorithm. However, the DAG-variant does not solve the computational problems in case a reasoning system does not output

minimal conflict sets (or close approximations of them). Therefore, we apply methods proposed in [12] to compute minimal conflict sets which allow us to use the original (and simpler) variant of Reiter's diagnosis methods.

For the computation of the HS-tree we employ a labeling that is similar to the original HS-tree. See Figure 1 for the HS-tree of our example. Nodes are labeled either by a minimal conflict set or by *consistent* ($\surd$). Closed branches are marked by $\times$. If a node n is labeled by a minimal conflict set $CS(n)$ then for each $s \in CS(n)$, edges are leading away from n which are labeled by s. The set of edge labels on the path leading from the root to node n is referred to as $H(n)$. If there does not exist a conflict set, the root is labeled by *consistent*. A node n must be labeled by a minimal conflict set CS if there exists a minimal conflict set CS s.t. $CS \cap H(n) = \emptyset$, otherwise this node is labeled by *consistent*.

The HS-tree is computed as follows by the application of pruning rules. The result is a pruned HS-tree which contains also closed branches. The HS-tree is a directed tree from the root to the leaves.

- If no diagnosis exists stop with exception. I.e. there is a $e^+ \in TC^+$ s.t. $e^+ \cup B \cup NE$ is inconsistent.
- Generate the HS-tree in breath-first order, level by level.
- Try to generate a minimal conflict set CS for the root node. Label the root with this conflict set, if such a conflict set exists. Otherwise, label the root with *consistent*. In this case return no fault was discovered.
- If a node n' (other than the root) has to be labeled:
  1. If a node n is labeled by *consistent* and $H(n) \subseteq H(n')$ close n', no successors are generated.
  2. If node n has been generated and $H(n) = H(n')$ then close n'.
  3. If there exists a node n labeled by $CS(n)$ s.t. $CS(n) \cap H(n') = \emptyset$ then reuse $CS(n)$ to label n'.
  4. Otherwise try to generate a *minimal* conflict set $CS(n')$ for n' s.t. $CS(n') \cap H(n') = \emptyset$. Label n' with this conflict set, if such a conflict set exists. Otherwise, label n' with *consistent*.

The leaf nodes of such a pruned HS-tree are either closed nodes or nodes labeled with *consistent*. Let n be a node labeled with *consistent* then $H(n)$ is a minimal diagnosis. Since the HS-tree is computed in breath-first order, minimal diagnoses are generated with increasing cardinality. Consequently, for the generation of all minimum cardinality diagnoses only the first level of the HS-tree has to be generated, where a node is labeled with *consistent*.

For the generation of minimal conflict sets we employ a simplified variant of QUICKXPLAIN [12] (i.e. no preferences are considered). QUICKXPLAIN takes as inputs two sets of sentences. The first set is a knowledge base (KB) and the second set is a background theory (B). If the knowledge base joined with the background theory is consistent QUICKXPLAIN outputs *consistent*. If the background theory is inconsistent the output is $\emptyset$. Otherwise, the output is a minimal conflict set $CS \subseteq KB$ (w.r.t. a background theory). QUICKXPLAIN operates on a divide and conquer strategy where a sequence of calls to a consistency checker is performed in order to minimize the conflict

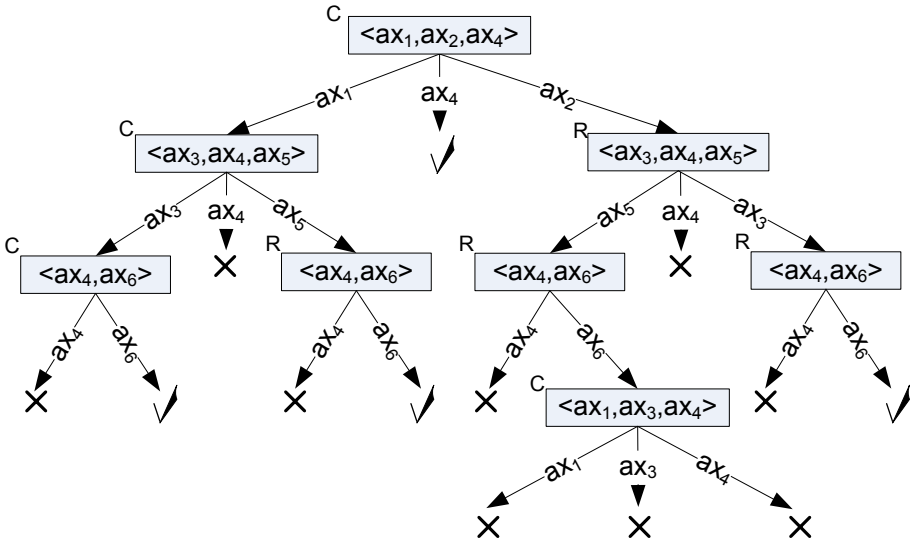

Fig. 1. HS-tree for the example given in Section 3. Closed branches are marked with $\times$. Computed minimal conflict sets are marked with C. Reused minimal conflict sets are marked with R. Consistent nodes are marked with $\surd$.

sets. If this divide and conquer strategy splits the knowledge base in half, QUICKXPLAIN needs $log_2(n/k) + 2k$ calls in the best case and $2k \cdot log_2(n/k) + 2k$ in the worst case where k is the length of the minimal conflict and n is the number of axioms in the knowledge base.

In order to generate a minimal conflict set for a node n, we have to check if there exists an $e^+ \in TC^+$ s.t. $(KB - H(n)) \cup B \cup e^+ \cup NE$ is inconsistent (i.e. $KB - H(n)$ contains a conflict set). For the calls to QUICKXPLAIN $KB - H(n)$ plays the role of the knowledge base and $B \cup e^+ \cup NE$ is considered as background theory. Note that an e^+ which is consistent with $(KB - H(n)) \cup B \cup NE$ need not be reconsidered for any successor n' of n since $H(n) \subset H(n')$. Therefore, we store for each node n all e^+ which were found to be consistent with $(KB - H(n)) \cup B \cup NE$ in the set $CE(n)$. For the generation of a label for n' we only need to check if there is an $e^+ \in \{TC^+ - \cup_{m \in predecessor(n')} CE(m)\}$ which is inconsistent with $(KB - H(n')) \cup B \cup NE$. The correctness and completeness of the generation of minimal diagnoses follows by the correctness and completeness of the HS-tree algorithm, QUICKXPLAIN, and the consistency checker.

Many factors are influencing the execution time of computing minimal diagnoses. The critical task of computing minimal conflict sets is dominated by the costs of consistency checking which strongly depend on the knowledge representation language as well as the actual content of the knowledge base. Finding a minimal diagnosis corresponds to a depth first construction of the HS-tree and therefore $|MD| + 1$ calls to QUICKXPLAIN are needed where $|MD|$ is the cardinality of this minimal diagnosis. However, we can construct cases where even the number of minimum cardinality di-

agnoses grows exponential in the problem size. Therefore, in practice the problem is simplified.

Diagnosis and conflict generation is exploited to guide further discrimination and repair actions. Therefore, only a set of *leading diagnoses* is generated which is a trade off between computational costs and further costs for diagnoses discrimination. Such actions may comprise additional tests, validation of axioms, and incremental repair. The definition of leading diagnoses is problem specific, e.g. a subset of minimal cardinality diagnoses. If necessary, the knowledge engineer can interrupt the generation of minimal diagnoses at any time and exploit the minimal conflicts and (partial) diagnoses found so far for further actions [13].

The execution time strongly depends on the actual diagnosis problem. In particular, computing minimal diagnoses (i.e. the HS-tree construction) significantly depends on the cardinality of the minimal diagnoses, the cardinality of minimal conflict sets, their reuse for constructing the HS-tree, and the actual costs of consistency checking. We therefore conducted various experiments in order to evaluate the execution time behavior for frequently used test ontologies.

6 Evaluation

The algorithms described above are implemented in JAVA (Version 1.5.01). For the consistency (coherence) checks we employed RACER (Version 1.7.23). The tests were performed on a PC (Intel Pentium M 1.8 GHz) with 1 GB RAM. The operating system was Windows XP Prof SP2. The results of our tests are depicted in Table 1. For these tests we employed the test knowledge bases bike2 to bike9, bcs3, galen, and galen3 provided at RACER's download site.[2]

For each test we randomly altered the knowledge bases. The result of each single alteration is an incoherent knowledge base. In order to introduce an incoherency we randomly picked two concepts where one concept subsumes the other (exploiting the taxonomy). In a next step, axioms which define these concepts were extended such that disjointness of these two concepts is enforced. An incoherent concept is the result. Consequently, every alteration will introduce at least one conflict set. However, since the introduction of these conflict sets is randomly performed there might be more but also less *minimal* conflict sets than the number of alterations.

The diagnosis task is to find minimal diagnoses in order to restore coherence. We did not provide a background theory and negative test cases because this corresponds just to additional axioms for consistency checks. The number of axioms (ax) for each knowledge base (including alterations) is stated in Table 1.

In order to provide realistic test cases from an application point of view we define a set of leading diagnoses. This set of leading diagnoses comprises the set of minimum cardinality (MC) diagnoses where we consider at most 10 diagnoses.

Note that in the worst case even the output of a single minimal conflict supports further actions for localizing faulty axioms. However, the generation of additional (minimal) diagnoses reduces the costs of actions for diagnoses discrimination and repair.

[2] http://racer-systems.com/products/download/index.phtml

We therefore not only measured the total time for computing leading diagnoses (TT) and the total time for performing coherence checks (COT) but also the total time for discovering the first minimal conflict (FCT) and the first minimum cardinality diagnosis (FDT). Time is measured in seconds. In addition to time information we reported the number of axioms contained in minimum cardinality diagnoses (|D|), the number of minimum cardinality diagnoses (# D, at most 10), the number of QUICKXPLAIN calls (QX), the number of coherence checks (# CH), and the number of minimal conflicts (# C) computed by the algorithm in order to compute the leading diagnoses for each test case. Since the cardinality of the minimal conflict sets defines the branching of the HS-tree we reported the minimum cardinality (min|C|) as well as the maximum cardinality (max|C|) of these conflict sets.

For each knowledge base we performed 30 tests. Each test corresponds to 4 random alterations (i.e. 8 changes) in order to evaluate the algorithms for multiple failure scenarios. Table 1 shows the average values of the test results as well as the data for the test case with minimum TT and maximum TT. Note, that for these special test cases the data values may lie above or below the average case.

The algorithm correctly computes the necessary conflicts. As expected each minimal conflict contains two changed axioms (beside others). All computed diagnoses are correct minimum cardinality diagnoses. Furthermore we empirically checked the completeness of the set of minimum cardinality diagnoses. As expected, the execution time greatly depends on the number and costs of the consistency checks. The costs of consistency checks not only depend on the number of axioms but on the content of a knowledge base. E.g. let us compare the maximum time cases of bike9 and bcs3. Although bcs3 is two times larger than bike9 and we require roughly 4 times more coherence checks (# CH) for bcs3 the time spent for these checks (COT) is almost the same.

As mentioned in the previous section the execution time for finding minimal diagnoses depends on the actual diagnosis problem. E.g. knowledge bases with many failures result in deep HS-trees whereas knowledge bases with many dependencies between the axioms result in high cardinality minimal conflict sets. These conflict sets cause broad HS-trees. The generation of conflicts and diagnoses shows no irregularity except for the knowledge base bcs3, where the cardinality of the minimal conflicts may become large, i.e. there are many axioms contributing to an incoherence because of the high cyclical complexity. As expected the overall execution time increases. However, discovering the first minimal conflict takes approximately a second for bcs3. Note, that the output of minimal conflict sets is already a valuable help for debugging the knowledge base. Even for the galen knowledge bases (approximately 4000 axioms) computing the first minimal conflict set takes not longer than 50 seconds.

In addition, we can observe that in the average, discovering the first minimum cardinality diagnosis requires roughly 80% of the total execution time. Therefore, spending some additional computational resources after the discovery of the first minimum cardinality diagnosis may be appropriate. At this stage the reuse of minimal conflict sets saves computational costs significantly.

The execution time behavior of the proposed method can be regarded as very satisfying given the size of the knowledge bases. Without such a support, debugging becomes

a very time consuming activity (e.g. locating multiple faults in hundreds or even thousands of axioms). Consequently, our tests show the practical applicability and utility of the proposed methods.

The integration of the consistency checker and QUICKXPLAIN is a source for improvements. If a consistency checker *efficiently* returns a set of axioms (i.e. a conflict, not necessarily minimal) involved in the generation of an inconsistency (incoherence) then this helps QUICKXPLAIN to reduce the number of consistency (coherence) checks. We recommend to implement this feature in consistency (coherence) checkers.

7 Related work

Diagnosis is strongly related to the generation of explanations. In the description logic community the work on explanations was pioneered by [14] and further enhanced for tableaux-based systems [15]. The intention of this work is to provide the basis for "natural" explanations of subsumption inferences. In particular, their goal is to derive a sequence of rule applications (i.e. proof fragments) which serve as a basis for natural explanations. Our approach is different since we compute minimal diagnoses which can be regarded as sources for unwanted behavior. We think that the work in the area of generating understandable proofs can be excellently integrated in a diagnosis framework for the explanations of conflicts (e.g. why a set of axioms is inconsistent).

In the area of description logics, the work by [4] is most closely related to our methods. However, we generalize and unify their concepts with concepts of the theory of diagnosis. Compared to our approach [4] require unfoldable $\mathcal{ALC}$-TBoxes. Their computation methods are based on the construction of tableaus where formulas are labeled. This label holds the information which axioms are relevant for the closure of branches. In contrast to this approach, our proposal works for arbitrary reasoners. However, provided that the label generation is not too expensive, we can explore this label for limiting the number of consistency (coherence) checks in order to speed up the computation.

In the work of [5] simple debugging cues are proposed which are integrated in an ontology development environment based on Pellet (open-source OWL DL reasoner). The main focus of their work is to improve the interaction between the knowledge engineer and the ontology development systems by debugging features. Regarding diagnosis our approach adds functionality, since we provided a clear definition of diagnosis (which allows the incorporation of test cases) and the correct and complete computation of multi-fault diagnoses.

Additional important work on improving the quality of ontologies is performed by [16] and [17]. The basic idea of these approaches is to find general rules and guidelines which assess the quality of ontologies. Furthermore, properties are expressed which specify conditions which must hold for error free ontologies. Some of these conditions may be formulated as test cases, but there are conditions which require reasoning about the terminology. This is beyond the expressive power of most ontology languages and therefore cannot be specified as tests. However, one possible extension which could be investigated is to generate a logical description of an ontology and to apply the general diagnosis approach to this description.

Since our method deals with the diagnosis of descriptions, the work on model-based diagnosis of hardware designs [13, 18] and software [19] shows some similarities. However, the fundamental difference is that these approaches have to generate a (logical) model of the description whereas in our domain we can exploit the descriptions directly.

8 Conclusions

In this paper we have proposed a general diagnosis theory for a broad range of ontology description languages. These concepts allow the formulation of test cases and the diagnosis of arbitrary knowledge bases containing terminological and assertional axioms. Minimal diagnoses identify minimal changes of the knowledge base such that the requirements specified by test cases can be met. We have provided algorithms which are correct and complete regarding the generation of all minimal diagnoses. Our methods are broadly applicable since they operate with arbitrary reasoning frameworks which provide consistency (coherence) checks. The practical feasibility of our method was shown by extensive test evaluations.

Acknowledgments

We thank anonymous referees for valuable remarks. The research project is funded partly by grants from the Austrian Research Promotion Agency, Programm Line FIT-IT Semantic Systems (www.fit-it.at), Project AllRight, Contract 809261 and the European Union, Project WS-Diamond, Contract 516933.

References

1. Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Scheider, P., Stein, L.: OWL Web Ontology Language Reference. W3C Recommendation, available at http://www.w3.org/TR/2004/REC-owl-ref-20040210/. (2004)
2. Horrocks, I., Patel-Schneider, P.: Reducing OWL entailment to description logic satisfiability. J. of Web Semantics **1** (2004) 345–357
3. Haarslev, V., Möller, R.: High performance reasoning with very large knowledge bases: A practical case study. In: Proc. IJCAI 01, Seattle, WA, USA (2001) 161–168
4. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. IJCAI 03, Acapulco, Mexico (2003) 355–362
5. Parsion, B., Sirin, E., Kalyanpur, A.: Debugging owl ontologies. In: WWW 2005, Chiba, Japan, ACM (2005)
6. Borgida, A.: On the relative expressive power of description logics and predicate calculus. Artificial Intelligence **82** (1996) 353–367
7. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
8. Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artificial Intelligence **152** (2004)

9. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **23** (1987) 57–95
10. Saraswat, V.A., de Kleer, J., Raiman, O.: Critical Reasoning. In: Proc. IJCAI 93. (1993) 18–23
11. Greiner, R., Smith, B.A., Wilkerson, R.W.: A correction to the algorithm in Reiter's theory of diagnosis. Artificial Intelligence **41** (1989) 79–88
12. Junker, U.: QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems. In: Proc. AAAI 04, San Jose, CA, USA (2004) 167–172
13. Friedrich, G., Stumptner, M., Wotawa, F.: Model-based diagnosis of hardware designs. Artificial Intelligence **111** (1999) 3–39
14. McGuinness, D.: Explaining Reasoning in Description Logics. PhD thesis, Department of Computer Science, Rutgers University (1996)
15. Borgida, A., Franconi, E., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F.: Explaining ALC subsumption. In: International Workshop on Description Logics, CEUR Workshop Proc. (CEUR-WS.org). Volume 22. (1999)
16. Guarino, N., Welty, C.: Evaluating Ontological Decisions with Ontoclean. Communications of the ACM **45** (2002) 61–65
17. Gómez-Pérez, A., Suárez-Figueroa, M.C.: Results of Taxonomic Evaluation of RDF(S) and DAML+OIL ontologies using RDF(S) and DAML+OIL Validation Tools and Ontology Platforms import services. In: Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools, CEUR Workshop Proc. (CEUR-WS.org). Volume 87. (2003)
18. Wotawa, F.: Debugging VHDL designs: Introducing multiple models and first empirical results. Applied Intelligence **21** (2004) 159–172
19. Chen, R., Wotawa, F.: Exploiting alias information to fault localization for Java programs. In: International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA2004), Gold Coast, Australia (2004)

KB		|D|	# D	# C	min |C|	max |C|	QX	# CH	FDT	FCT	COT	TT
bike2	min	3	6	4	2	6	10	134	23	7,1	27,4	35
154 ax	avg	3,7	8,7	5	2,1	4	13,7	181	47,5	8,3	44,2	55
	max	4	10	7	2	4	17	284	61,4	10,1	62,3	77
bike3	min	4	10	4	3	3	14	120	16,3	3,4	16,2	19
109 ax	avg	4,5	9,2	5,6	2,6	3	14,9	164	22,4	3,4	25,5	29
	max	4	6	7	2	3	13	202	22,4	4,3	31,7	37
bike4	min	3	10	4	3	4	14	162	52,3	12,5	53,1	58
166 ax	avg	3,6	9,6	5,9	2,6	5	15,5	244	71,1	12,1	76,7	84
	max	4	10	8	3	10	18	358	83,3	13,4	104	115
bike5	min	1	1	3	3	4	4	131	40,6	20,1	56	60
184 ax	avg	2,6	5,9	4,6	2,9	3,9	10,5	193	79,8	22	97,4	105
	max	3	7	6	3	4	13	247	90,7	21,6	135	145
bike6	min	1	1	3	3	4	4	137	54,3	26,3	75	80
207 ax	avg	3	7	5,1	2,8	4	12,1	220	108,5	25,3	127	135
	max	3	7	6	3	4	13	263	111,2	25,5	160	171
bike7	min	1	2	2	3	3	4	84	12,6	11,6	23,1	25
162 ax	avg	2,9	8,3	3,6	2,8	3	11,9	151	40	12,2	49,9	54
	max	3	8	5	2	3	13	186	57,3	12,9	67,7	73
bike8	min	2	4	3	2	3	7	104	33,7	17	50,4	54
185 ax	avg	3,2	8,9	4	2,7	3	12,9	172	59,7	17	72,7	79
	max	4	10	5	3	3	15	216	89,9	16,3	91,5	99
bike9	min	1	1	3	3	4	4	127	50,1	23,3	72,8	78
215 ax	avg	3,1	7,2	4,9	2,7	4	12,1	211	116,2	27,1	131	140
	max	4	10	5	3	4	15	218	242,6	28,5	243	253
bcs3	min	3	4	4	2	3	8	118	16,3	1	15,5	18
432 ax	avg	3,4	7,1	5,7	2	17,1	12,9	276	46,7	1	51,4	61
	max	4	10	9	2	51	19	968	251,7	1,2	232	269
galen	min	2	2	3	2	2	5	86	95,8	30,4	65,4	104
3963 ax	avg	2,3	3,1	3,2	2	2	6,4	104	172,4	41	125	227
	max	3	8	3	2	2	11	116	223,6	39,8	234	366
galen3	min	1	1	2	2	2	3	53	60	49,6	38,2	105
3927 ax	avg	2,2	3,6	3	2	2	6,6	94,7	157	34,4	93,2	203
	max	4	10	4	2	2	14	150	452,1	40,3	421	489

Table 1. Test results for diagnosing faulty knowledge bases. Columns are: number of axioms contained in minimum cardinality diagnoses (|D|), number of minimum cardinality diagnoses (# D, at most 10), number of minimal conflict sets computed (# C), cardinality of smallest minimal conflict set (min|C|), cardinality of largest minimal conflict set (max|C|), number of QUICKXPLAIN calls (QX), number of coherence checks (# CH), total time for discovering the first minimum cardinality diagnosis (FDT), total time for discovering the first minimal conflict set (FCT), total time for performing coherence checks (COT), total time for computing leading diagnoses (TT). Time is measured in seconds.