Diagnosing Terminologies

Stefan Schlobach
Department of Computer Science
Vrije Universiteit Amsterdam
The Netherlands
Email: schlobac@few.vu.nl

Abstract

We present a framework for the debugging of logically contradicting terminologies, which is based on traditional model-based diagnosis. To study the feasibility of this highly general approach we prototypically implemented the hitting set algorithm presented in (Reiter 1987), and applied it in three different scenarios. First, we use a Description Logic reasoning system as a black-box to determine (necessarily *maximal*) conflict sets. Then we use our own non-optimized DL reasoning engine to produce *small*, and a specialized algorithm to determine *minimal* conflict sets. In a number of experiments we show that the first method already fails for relatively small terminologies. However, based on small, or minimal conflict sets, we can often calculate diagnoses in reasonable time.

Introduction

Ontologies are widely used in a variety of different applications, and the recent past has seen a surge in modeling tools to support the creation of high quality ontologies. What is usually missing is diagnostic support when they are logically incoherent. Recent approaches have focused on debugging of terminologies, and were only applicable to particular logical representation, and restricted formalisms.

Incoherence can have several causes. Modeling errors occur because constructing an ontology is a very difficult process, and the complexity of both the problem and the representation languages easily leads to logical contradictions. Alternatively, incoherence is often a result of migration or merging of ontologies. (Cornet & Abu-Hanna 2002) describes how to create Description Logic (DL) terminology from a frame-based representation. For the migration from frames to DL modeling decisions have to be taken to interpret the frame semantics in DL, and a stringent migration can lead to a high number of unsatisfiable concepts. Similar problems arise when two or more ontologies are merged, for example in the context of the Semantic Web. Using both SUMO and CYC (two upper ontologies) in a single document leads to over 1000 unsatisfiable concepts.

Schlobach & Cornet were among the first to develop a general framework for debugging of erroneous terminologies in (2003), and the authors provide a specialized algorithm for the Description Logic (DL) $\mathcal{ALC}$. They also discuss their experience with the debugging of DICE. Unfortunately, the proposed methods have two main shortcomings.

1. The minimal sub-terminologies considered are small error-containing terminologies, but they are not further analyzed to provide further information on which of the axioms should be ignored or fixed.
2. More serious, however, is the restriction to $\mathcal{ALC}$, and the need for a full reimplementation of a DL tableau engine.

We try to overcome these problems by placing terminological debugging in the context of model-based diagnosis (MBD). In his seminal paper, Reiter (1987) introduced *diagnoses* as smallest sets of components that need fixing to render a system (represented as a set of first-order formulas) correct. He provides a generic method to calculate diagnoses on the basis of conflict sets and their minimal hitting sets.

Terminological debugging is a special case of MBD, and we use Reiter's algorithms to calculate diagnoses for erroneous terminologies. These diagnoses will be minimal sets of terminological axioms that have to be ignored in order to turn the terminology coherent. Based on Reiter's hitting set tree (HS-tree) algorithm we have an optimized way of calculating diagnoses, where we can use available DL reasoner as a black-box. This has the huge advantage that the expressiveness of the terminologies to be debugged, is only restricted by the expressiveness of the reasoner itself.

The computational price for this gain in expressiveness is very high, though. Based on a prototypical implementation we evaluated and compared three different types of diagnoses, all based on Reiter's original HS-tree method. The general idea is that diagnoses are paths in minimal trees, where each of the nodes is labeled with a set of contradicting axioms (the conflict sets), and where each edge on the path "hits" precisely one node label on its way to the leaves.

The algorithmic difference in our three approaches is in the choice of conflict sets. First, we use RACER (Haarslev & Möller 2001) as a black-box to return entire terminologies as maximal conflict sets. This approach, though the most general one, fails even on small incoherent terminologies. The second idea uses internal information from unsatisfiability proofs. As such information is not available from any of the current DL reasoner, we implemented a (non-optimized) tableau prover for $\mathcal{ALC}$ satisfiability for unfoldable TBoxes, which returns small (though not minimal) sets of axioms

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

contributing to the closure of the tableau. Finally, the third approach implements the specialized algorithms described in (Schlobach & Cornet 2003) to calculate minimal incoherence preserving sub-terminologies as minimal conflict sets.

In a number of experiments with publicly available terminologies we evaluate the feasibility of diagnosis, in particular studying the effect of using a general method versus more specialized algorithms. The outcome is clear: only for very small examples the general method works in reasonable time, and at least small conflict sets are required to produce some diagnoses more or less efficiently.

Related work

How to build good ontologies has been discussed extensively in the literature, and many links can be found on the W3C website about methodology and languages. For Description Logics the handbook (Baader *et al.* 2003) is an excellent reference. Explanation has been an issue in the DL community for several years, but most papers, such as (Borgida, Franconi, & Horrocks 2000), deal with subsumption. The number of papers dealing with theoretical studies of reasoning with inconsistency is enormous (to mention (Beziau 2000; Schaerf & Cadoli 1995) or (Huang, van Harmelen, & ten Teije 2005) from a DL perspective). The only work on the detection and explanation of incoherences we are aware of is (Schlobach & Cornet 2003) that we mentioned earlier. From a more practical point of view, closest to our work are the Chimaera and PROMPT tools described in (McGuinness *et al.* 2000) and (Noy & Musen 2000), which provide support for merging and analysis of knowledge bases but not for debugging.

The literature on model-based diagnosis is manifold, but we focus on the seminal (Reiter 1987), and (Greiner, Smith, & Wilkerson 1989), which corrects a small bug in Reiter's original algorithm. We refer the interested reader to a good overview in (Console & Dressler 1999).

Diagnosing Description Logic Terminologies

In this section we will show how to represent the debugging of Description Logic ontologies[1] as a model-based diagnosis problem. Description Logics are a family of well-studied set-description languages which have been in use for over two decades to formalize knowledge. They have a well-defined model theoretic semantics, which allows for the automation of a number of reasoning services.

We shall not give a formal introduction into Description Logics here, but point to the second chapter of the DL handbook (Baader *et al.* 2003). Briefly, in DL concepts will be interpreted as subsets of a domain, and roles as binary relations. In a terminological component $\mathcal{T}$ (called TBox) the interpretations of concepts can be restricted to the *models* of $\mathcal{T}$. Let, throughout the paper, $\mathcal{T} = \{Ax_1, \ldots, Ax_n\}$ be a set of axioms, where Ax_i is of the form $C_i \dot{\sqsubseteq} D_i$ for each $1 \leq i \leq n$ and arbitrary concepts C_i and D_i.

Let $\mathcal{U}$ be a finite set, called the universe. A mapping $\mathcal{I}$, which interprets DL concepts as subsets of $\mathcal{U}$ is then a *model* of an axiom $C \dot{\sqsubseteq} D$, if, and only if, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Based on this semantics a TBox can be checked for *incoherence*, i.e., whether there are *unsatisfiable* concepts: concepts which are necessarily interpreted as the empty set in all models of the TBox. More formally

1. A concept A is *unsatisfiable* w.r.t. a terminology $\mathcal{T}$ if, and only if, $A^{\mathcal{I}} = \varnothing$ for all models $\mathcal{I}$ of $\mathcal{T}$.
2. A terminology $\mathcal{T}$ is *incoherent* if there exists a concept-name in $\mathcal{T}$, which is unsatisfiable.

Conceptually, these are *simple* modeling errors because we assume that a knowledge modeler would not specify an empty concept in a complex way.

In (2003) Schlobach & Cornet propose to explain unsatisfiability by minimal sets of axioms contributing to the unsatisfiability (similarly for incoherence). To illustrate their approach they introduce the (incoherent) TBox $\mathcal{T}^*$, with primitive concepts A, B and C and defined concepts $A_1, \ldots, A_7$ and roles r and s:

$ax_1{:}A_1 \dot{\sqsubseteq} \neg A \sqcap A_2 \sqcap A_3$	$ax_2{:}A_2 \dot{\sqsubseteq} A \sqcap A_4$
$ax_3{:}A_3 \dot{\sqsubseteq} A_4 \sqcap A_5$	$ax_4{:}A_4 \dot{\sqsubseteq} \forall s.B \sqcap C$
$ax_5{:}A_5 \dot{\sqsubseteq} \exists s.\neg B$	$ax_7{:}A_7 \dot{\sqsubseteq} A_4 \sqcap \exists s.\neg B$
$ax_6{:}A_6 \dot{\sqsubseteq} A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$	

Even for this simple TBox it is non-trivial to pinpoint to the core of the erroneous modeling. State-of-the-art DL reasoning tools efficiently detect, but fail to explain, unsatisfiability of the concept-names $\{A_1, A_3, A_6, A_7\}$.

Let A be a concept that is unsatisfiable in a TBox $\mathcal{T}$. A set $\mathcal{T}' \subseteq \mathcal{T}$ is a *minimal unsatisfiability-preserving sub-TBox (MUPS)* of $\mathcal{T}$ if A is unsatisfiable in $\mathcal{T}'$, and A is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$. The set of MUPS for TBox $\mathcal{T}^*$ and, for example, its unsatisfiable concept A_1 is $\{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$. Incoherence can be explained using the smallest subsets of an original TBox preserving unsatisfiability of at least one atomic concept. An incoherent 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. For $\mathcal{T}^*$ there are three MIPS: $\{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$. It can easily be checked that each of the three incoherent TBoxes in $mips(\mathcal{T}^*)$ is indeed a MIPS as taking away a single axiom renders each of the three coherent.[2]

Debugging as model-based diagnosis

In (1987), Ray Reiter introduced a general framework for diagnosis based on first principles. He defines a *system* as a

[1] Throughout the paper we will interchange the terms terminology and ontology. Formally, with a terminology we denote a set of terminological axioms, whereas an ontology can also contain assertional knowledge. In principle, the methods we describe work for ontologies as well as for terminologies, but to simplify matters, we mostly restrict our attention to terminologies.

[2] Schlobach & Cornet introduced a specialized algorithm based on Boolean minimization of axioms in a tableau proof for unfoldable $\mathcal{ALC}$-TBoxes, and report on their experience when calculating MIPS and MUPS to debug the DICE terminology. $\mathcal{ALC}$ is a simple yet relatively expressive DL with full negation and both universal and existential quantification. A TBox is called *unfoldable* if the left-hand sides of the axioms are atomic, and if the right-hand sides contain no direct or indirect reference to the defined concept (Nebel 1990).

pair (Sd, Cmp) where Sd, the *system description*, is a set of first order (FO) sentences, and where Cmp, the *system components*, is a finite set of constants. To represent a terminology as a system for terminological debugging we represent satisfiability and incoherence as first-order satisfiability. Let $(C, x)^t$ be the standard translation from a concept C into FO-logic given a variable x, i.e. where, for example,

$$\begin{aligned}(C_1 \sqcap C_2, x)^t &= (C_1, x)^t \wedge (C_2, x)^t,\\ (\neg C, x)^t &= \neg (C, x)^t,\\ (\exists R.C, x)^t &= \exists R(x, y) \wedge (C, y)^t,\\ (A, x)^t &= A(x) \text{ for atomic concept names.}\end{aligned}$$

This standard translation can be trivially extended to TBox axioms: $(C \dot{\sqsubseteq} D)^t = \forall x.(C, x)^t \rightarrow (D, x)^t$. A TBox $\mathcal{T}$ translates into the first-order statement $\mathcal{T}^t = ax_1^t \wedge \ldots \wedge ax_n^t$.

Terminological and first-order satisfiability have a different flavor. Consider the translation of example TBox $\mathcal{T}^*$.

$$\begin{aligned}&\forall x(A_1(x) \rightarrow \neg A(x) \wedge A_2(x) \wedge A_3(x)) \wedge\\ &\ldots\\ &\forall x(A_7(x) \rightarrow A_4(x) \wedge \exists y(S(x, y) \wedge \neg B(y)))\end{aligned}$$

which is consistent, even though A_1 is terminologically unsatisfiable. To make the translation satisfiability preserving, in the sense that a concept is satisfiable w.r.t. a TBox if, and only if, its first-order correspondence formula is satisfiable, we introduce *expectations*, such as $Ex = \{\exists y A(y)\}$. Then, A is satisfiable w.r.t. $\mathcal{T}$ if, and only if, $\mathcal{T}^t \cup E$ is satisfiable.[3]

Terminological system descriptions Terminological system description will capture the semantics of the terminology, and the components are those axioms, which are potentially erroneous. A distinct predicate $AB(\cdot)$ can be added to denote abnormality of components. In our interpretation, truth of this predicate means that the axiom is erroneous, and should not contribute to the semantics of the terminology. The *(terminological) system description* $Sd(\mathcal{T})$ for a terminology $\mathcal{T}$ is the FO-formula:

$$\left(\neg AB(ax_1) \rightarrow ax_1^t\right) \wedge \ldots \wedge (\neg AB(ax_n))$$

A diagnosis problem occurs when the terminological system description is unsatisfiable w.r.t. a set of expectations.[4]

Definition 1 Let $Sd(\mathcal{T})$ be terminological system description of $\mathcal{T}$, and Ex be a set of FO-formulas called *expectations*. Let, furthermore $Cmp \subseteq \mathcal{T}$ be a set of axioms, the *components*. We call $(Sd(\mathcal{T}), Ex, Cmp)$ a *(terminological) diagnosis problem* if $Sd(\mathcal{T}) \cup Ex)$ is inconsistent.

Let us look at two particular diagnosis problems, first, to explain unsatisfiability of a particular concepts, and,secondly, to explain incoherence. In what follows we will call the terminological diagnosis problem $(Sd(\mathcal{T}), \{\exists y A(y)\}, \mathcal{T})$ the *unsatisfiability problem*, and $(Sd(\mathcal{T}), \{\bigwedge_{A \in \mathcal{T}} \exists y A(y)\}, \mathcal{T})$ the *incoherence problem*.

[3]The distinction between first-order and DL satisfiability is well-known, but has sometimes lead to confusion in discussions with people without the DL background.

[4]Expectations replace observations in Reiter's original framework. Observations follow from the semantics of the system.

Terminological diagnosis We extend Reiter's definition of a diagnosis to terminological diagnosis problems by applying Proposition 3.4. of (Reiter 1987). Note, that in this framework there is no conceptual difference between diagnosing incoherence and diagnosing unsatisfiability problems.

Definition 2 A *(terminological) diagnosis* for $(Sd(\mathcal{T}), Ex, Cmp)$ is a minimal set $\Delta \subseteq \mathcal{T}$ such that

$$Sd(\mathcal{T}) \cup Ex \cup \{\neg AB(ax) \mid ax \in \mathcal{T} \setminus \Delta\} \text{ is consistent.}$$

In DL terms, a diagnosis Δ is a minimal sub-terminology of an unsatisfiable (or incoherent) terminology $\mathcal{T}$ (w.r.t. a concept A), such that A is satisfiable w.r.t. the remaining TBox $\mathcal{T} \setminus \Delta$ (respectively, that $\mathcal{T} \setminus \Delta$ is coherent).

From $mips(\mathcal{T}^*) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$ the 6 diagnoses $\{ax_1, ax_3, ax_7\}$, $\{ax_1, ax_4\}$, $\{ax_1, ax_5, ax_7\}, \{ax_2, ax_3, ax_7\}, \{ax_2, ax_5, ax_7\}$ and $\{ax_2, ax_4\}$ can be derived. It can easily be checked, that for all these $\mathcal{T}^\Delta$, the TBoxes $\mathcal{T}' = \mathcal{T} \setminus \mathcal{T}^\Delta$ are coherent, and that there are no smaller $\mathcal{T}^\Delta$ with this property.

It should be noted that diagnoses and MIPS (MUPS) are complementary for debugging. A diagnosis suggests which axioms should be ignored (or fixed) to make the terminology coherent, but not every diagnosis necessarily contains the erroneous axiom. Suppose, that ax_2 and ax_4 contain errors, and that all other axioms are correct. The first diagnose $\{ax_1, ax_3, ax_7\}$, though correct, will not identify the error. In a large diagnoses space it might be difficult to find the right diagnosis. Each MIPS, on the other hand, definitively contains a culprit for the logical conflict.

Calculating terminological diagnoses

Terminological diagnosis, as defined in the previous section, is an instance Reiter's diagnosis from first principles. Therefore, we can use Reiter's algorithms to calculate terminological diagnoses. What is required is a method to produce conflict sets, and we will discuss three different options for this. Let us first recall the basic methodology from (Reiter 1987).

Given an incoherent terminology $\mathcal{T}$, a *conflict set* for $(Sd(\mathcal{T}), Ex, Cmp)$ is a set $CS \subseteq Cmp$, such that $Sd(\mathcal{T}) \cup Ex \cup \bigcup_{ax \in CS} \{\neg AB(ax)\}$ is inconsistent. A conflict set is minimal if, and only if, no proper subset of it is a conflict set for the same diagnosis problem.

The following proposition is the basis for calculating diagnoses on the basis of conflict sets.

Proposition 1 ((Reiter 1987) Proposition 4.2.) A set $\Delta \subseteq \mathcal{T}$ is a diagnosis for a terminological diagnosis problem $(Sd(\mathcal{T}), Ex, Cmp)$ iff Δ is a minimal set such that $\mathcal{T} \setminus \Delta$ is not a conflict set of $(Sd(\mathcal{T}), Ex, Cmp)$.

The basic idea to calculate diagnoses from conflict sets is based on minimal hitting sets. Suppose C is a collection of sets. A *hitting set* for C is a set $H \subseteq \bigcup_{S \in C} S$ such that $H \cap S \neq \varnothing$ for each $S \in C$. A hitting set is minimal for C iff no proper subset of it is a hitting set for C.

This gives the basis of Reiter approach to calculate diagnoses given the following theorem which is a direct consequence of Corollary 4.5 in (Reiter 1987).

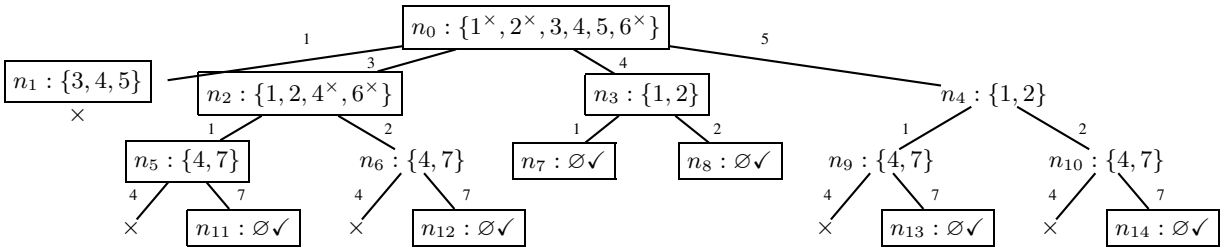

Figure 1: HS-Tree with small conflict sets (14 nodes & 11 calls to the DL reasoner)

Theorem 1 A set $\Delta \in \mathcal{T}$ is a diagnosis for a terminological diagnosis problem $(Sd(\mathcal{T}), Ex, Cmp)$ iff Δ is a minimal hitting set for the collection of conflict sets for $(Sd(\mathcal{T}), Ex, Cmp)$.

To calculate minimal hitting trees Reiter introduces hitting set trees (HS-trees). For a collection C of sets, a HS-tree T is the smallest edge-labeled and node-labeled tree, such that the root is labeled by $\checkmark$ if C is empty. Otherwise it is labeled with any set in C. For each node n in T, let $H(n)$ be the set of edge labels on the path in T from the root to n. The label for n is any set $S \in C$ such that $S \cap H(n) = \varnothing$, if such a set exists. If n is labeled by a set S, then for each $\sigma \in S$, n has a successor, n_σ joined to n by an edge labeled by σ. For any node labeled by $\checkmark$, $H(n)$, i.e. the labels of its path from the root, is a hitting set for C.

Figure 1 shows a HS-tree T for the collection $C = \{\{1, 2, 3, 4, 5, 6\}\{3, 4, 5\}, \{1, 2, 4, 6\}, \{1, 2\}, \{4, 7\}\}$ of sets. T is created breadth first, starting with root node n_0 labeled with $\{1, 2, 3, 4, 5, 6\}$. For diagnostic problems the sets in the collection are conflict sets which are created on demand. In our case, conflict sets for a terminological diagnosis problem can be calculated by a standard DL engine because of the following simple proposition.

Proposition 2 For any set C of components (terminological axioms) in a terminological diagnostic problem, the FO-formula $Sd(\mathcal{T}) \cup Ex \cup \bigcup_{ax \in C} \{\neg AB(ax)\}$ is inconsistent if, and only if, A is unsatisfiable in $\mathcal{T} \setminus C$.

These calls are computationally expensive, which means that we have to minimize them. In Figure 1, those nodes are boxed, for which labels were created by calls to the prover. T reuses already calculated and smallest possible labels, and is pruned in a variety of ways, which are defined in detail in (Reiter 1987). Just for example, node n_0 is relabeled with a subset $\{3, 4, 5\}$ of its label. We denote by $1^\times$, that element 1 is deleted. Note, that no successor for this element has to be created. Node n_6 has been automatically labeled with $\{4, 7\}$, because the intersection of its path $h(n_6) = \{2, 3\}$ is empty with an already existing conflict set in the tree.

Three ways of implementing diagnosis

The generality of Reiter's algorithm has the advantage of giving some leeway for particular methodological choices. We implemented three ways of calculating conflict sets.

1. Use an optimized DL reasoner to return a conflict set in each step of the creation of the HS-tree. The only way to get conflict sets for an incoherent TBox $\mathcal{T}$ is to return $\mathcal{T}$ itself, i.e. the *maximal conflict set*.
2. Use an adapted DL reasoner to return *small conflict sets*, which it can derive from the clashes in a tableau proof.
3. Use a specialized method to return *minimal conflict sets*, e.g., using the algorithms of (Schlobach & Cornet 2003).

Diagnosis with maximal conflict sets The most general way to calculate terminological diagnosis based on hitting sets is to use one of the state-of-the-art optimized DL reasoner. The advantage is obvious: the expressiveness of the diagnosis is only restricted by the expressiveness of the DL reasoning implemented in the reasoner. We use RACER, which allows to diagnose incoherent terminologies up to $\mathcal{SHIQ}$ without restriction on the structure of the TBox. The algorithm to use RACER is simple: if $\mathcal{T}$ is incoherent, return $\mathcal{T}$, other return $\varnothing$. As RACER is highly optimized we can expect to get the maximal conflict sets efficiently.

The disadvantage of this naive approach is that the conflict sets are huge, and even with reusing of node labels and pruning, the HS-tree become quickly to large to handle. Take TBox $\mathcal{T}^*$ with its incoherence problem $(Sd(\mathcal{T}^*), \{\bigwedge_{A \in \mathcal{T}} \exists y A(y)\}, \mathcal{T}^*, \varnothing)$, where related HS-tree already has 380 nodes, and needs 67 calls to RACER. We will see that the price we pay for the gain in expressiveness is too high, and that smaller conflict sets are required.

Diagnosis with small conflict sets The disadvantage of using a DL reasoner as a black-box is that they do not provide any information on which components contribute to the incoherence. Technically, this means which axioms contribute to the closure of the tableau. To show that already straightforward collecting of clash-enforcing axioms can dramatically improve the efficiency of diagnosis, we implemented a simple tableau calculus for unfoldable $\mathcal{ALC}$ TBoxes. This reasoner returns an unordered, and not necessarily minimal, list of axioms which are (indirectly) responsible for the clashes in the tableau. The basic idea is to label each formula with a set of axioms, which are added to a formula in the tableau whenever they are used to "unfold" a defined concept. This algorithm is not optimized, but returns small conflict sets, and the sizes of the HS-Trees decrease dramatically. Figure 1 shows the hitting tree for the incoherence problem for $\mathcal{T}^*$ where small conflict sets have been collected from tableau proofs. Compared to the previous method, there were only 14 nodes created, and 11 calls to the DL reasoner necessary.

	#ax	#unsat	#mips	|mips|	length of mD	RACER timeCC	Maximal CS #D/hr	Maximal CS timeD1	Small CS #D/hr	Small CS timeD1	Minimal CS #D/hr	Minimal CS timeD1
DICE-A	534	76	16	3	3	88 s	0	-	4	1622 s	27	151 s
MGED	406	72	38	4	3	3 s	0	-	10	40 s	58	31 s
Geo	417	11	22	2.6	8	1.3 s	-	-	8	114 s	115	62 s
S&C	6382	923	-	-	-	45 s	0	-	0	-	0	-
WINE	176	10	-	-	2	1 s	6	37 s	-	-	-	-
MadC	69	1	-	-	1	0.4 s	4	12 s	-	-	-	-
	1	2	3	4	5	6	7	8	9	10	11	12

Table 1: Comparing Diagnosis with different types of conflict sets

Diagnosis with minimal conflict sets Previously, we recalled the notion of minimal unsatisfiability (and incoherence) preserving sub-terminologies MUPS and MIPS, which were introduced in (Schlobach & Cornet 2003) for the debugging of terminologies.

The MUPS of an incoherent terminology $\mathcal{T}$ and an unsatisfiable concept A are the minimal conflict sets for the unsatisfiability problem $(Sd(\mathcal{T}), \{\exists y A(y)\}, \mathcal{T}, \varnothing)$. It is easily checked that each MUPS $\{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$ for A_1 and $\mathcal{T}^*$ is indeed a minimal conflict set for the unsatisfiability problems $(Sd(\mathcal{T}^*), \exists y A_1(y)), \mathcal{T}^*, \varnothing)$. This time, only 12 nodes were created. Based on the MUPS, it is straightforward to calculate MIPS, which are the minimal conflict sets for the incoherence problem.

Proposition 3 The MIPS of an incoherent terminology $\mathcal{T}$ are the minimal conflict sets for the incoherence problem $(Sd(\mathcal{T}), \{\bigwedge_{A \in \mathcal{T}} \exists y A(y)\}, \mathcal{T}, \varnothing)$.

Experiments

With a number of experiments we studied the feasibility of diagnosis. We implemented the three techniques described in the previous section in JAVA,[5] and applied them to a number of publicly available Description Logic terminologies.

We split our test terminologies in three groups, ordered by how they were built. As an example for a terminology created through migration we consider a previous version of the anatomy fragment of DICE (we abbreviate DICE-A), with 534 axioms and 76 unsatisfiable concepts. The incoherence of DICE-A has two distinct causes: first, this is a snap-shot from the terminology in its creation process, i.e. it contains real modeling errors. Moreover, the high number of contradictions is specific for migration as a result of stringent semantic assumptions. MGED and Geo are variants of ontologies which are incoherent because they have disjointness statements artificially added for semantic enrichment (as suggested in (Schlobach 2005)). MGED provides standard terms for the annotation of micro-array experiments to enable structured queries on those experiments; and Geo an ontology of geography made available by the Teknowledge Corporation. The third category contains the merged ontology of SUMO and CYC, two well-known upper ontologies. As they are topic-related, and as CYC provides disjointness statements, there is a high number of unsatisfiable concepts.

[5]Implementations and test sets will be made publicly available.

We constructed simplified $\mathcal{ALC}$ versions for all five terminologies. Without loss of unsatisfiability, we removed, for example, numerical constraints, role hierarchies and instance information. All terminologies, however, were non-cyclic and could be transformed to an unfoldable format.

For the last two examples, WINE and MadC, this is not the case. They were constructed to illustrate language features of Description Logics, and we use them to illustrate Reiter's generic method works for expressive formalisms, where both other methods fail. MadC is incoherent with unsatisfiable concept *MadCow*, but we have enriched the wine terminology with five erroneous statements, resulting in a terminology with 10 unsatisfiable concepts.

Quantitative analysis: Table 1 summarizes the quantitative results of diagnosis on the 6 incoherent terminologies introduced above. All experiments were performed on a Pentium III, 1.3.GHz, RedHat Linux. The first 4 columns summarize information about the terminologies, the number of axioms, unsatisfiable concepts and MIPS, as well as the average size of the MIPS. Column 5 gives the length of the smallest diagnosis, Column 6 the time RACER needs to check for incoherence of the terminology. The diagnostic results are split in three pairs: the first two columns 7 and 8 ((labeled Maximal CS) give

- the number of diagnoses calculated per hour (#D/hr), and
- the time to calculate the first diagnosis (timeD1)

based on maximal conflict sets. Similarly for columns 9 and 10, for small, and 11 and 12 for minimal conflict sets (calculated using MIPS). The runtime in column 12 contains calculation of unsatisfiability using RACER, the calculation of the MIPS, and, finally, of the hitting sets.

Qualitative analysis: The most significant result is the almost complete failure to calculate diagnoses using the naive maximal hitting set approach. Only for the toy examples of the WINE and MadC terminologies are any diagnoses found. The reason for this is the length of the minimal diagnosis and the number of axioms. As all but one axioms belong to the maximal conflict sets, there are (#ax-1) branches at first level, and (#ax-1)*(#ax-2) branches at level 2. To find a diagnosis of length 3, a branch of depth 3 has to be explored, which means, e.g., for DICE-A a total number of 100 million branches. Only small ontologies with small diagnoses can be debugged in this most general way.

Things look better for the other methods. Both detect diagnoses of size up to 8 for large terminologies such as DICE-A or GEO. Again, all depends on the size of the diagnoses and the number of axioms. Still the results were unsatisfactory: there was not a single algorithm that determined any diagnoses for the merged SUMO and CYC ontology, and only once did the algorithm terminate within an hour, namely when checking DICE-A using minimal conflict sets.

For this latter method (based on minimal conflict sets) the computational difficulty lies in the fact that constructing minimal hitting sets from MIPS corresponds to calculating prime implicants for a propositional formula, and is thus an NP-COMPLETE problem. Although our prototypical implementation uses some optimization it is not efficient enough to build and search very large HS-Trees. The intermediate implementation based on small conflict sets could significantly be made faster by using an optimized reasoner to return conflict sets more efficiently. From manual inspection we believe that the size of the conflict sets (and thus the size of the HS-Tree) would not be much smaller, but the time to find the small conflict sets could be significantly lower. In both cases, however, the theoretical (and practical) complexity is very high, so that we doubt that terminologies such as the merged S&C can be diagnosed in the near future.

Conclusion

We present a model-based diagnosis approach for the debugging of Description Logic terminologies. Representing the terminological incoherence problems as a first-order system descriptions allows us, at least in theory, to use Reiter's general framework to calculate diagnoses for very expressive terminologies. There are several conceivable extensions: one can use diagnoses to explain correct and incorrect subsumption. Given a coherent terminology, one can specify subsumption (or non-subsumption) as expectations. Similarly, extending diagnosis with instances is easy, one simply has to add assertional axioms to the system description, and instance relations to the expectations. In all cases, diagnosis works "off-the-shelf" as long as the components are set of axioms. But even this leaves room for extension, as one can easily choose particular subsets of a terminology as sets of components, which not only can be very useful in practice, but can improve the efficiency significantly.

This will be necessary, as our experimental evaluation shows that the complexity of the general problem is so high that it is doubtful whether it will work on large incoherent terminologies. Only the two more specialized algorithms work in practice, which implies that implicit information on proofs is required. We believe that it should be feasible to extract small conflict sets from a more verbatim output of current DL reasoner. Applying an optimized, efficient and expressive general-purpose reasoning system for small conflict set creation has many advantages, as it would make the much more efficient method based on small conflict sets available for terminologies such as WINE or MadC.

We are currently implementing an interesting alternative, which is to calculate MIPS in a generic, bottom-up way. The idea is to create sub-TBoxes of an incoherent terminology of increasing size, and check for correctness using RACER. This combines the advantages of the three approaches described in this paper; we have a generic algorithm for debugging of incoherent terminologies, which should be reasonably efficient, and applicable on expressive ontologies. An evaluation of this new algorithm and the methods described in (Schlobach & Cornet 2003) is forthcoming.

These remarks also determine our future research directions: first we will address the computational problems by improving the HS-Tree implementation as well as the tableau calculus of our DL engine. The next step will be to stepwise extend the expressiveness of the diagnosable terminologies and the functionality by explaining subsumption and instance relations as described above.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook*. Cambridge University Press.

Beziau, J.-Y. 2000. What is paraconsistent logic? In Batens, D.; Mortensen, C.; Priest, G.; and Van Bendegem, J., eds., *Frontiers of paraconsistent logic*. 95–111.

Borgida, A.; Franconi, E.; and Horrocks, I. 2000. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence*, 209–213.

Console, D., and Dressler, O. 1999. Model-based diagnosis in the real world: Lessons learned and challenges remaining. In *IJCAI*, 1393–1400.

Cornet, R., and Abu-Hanna, A. 2002. Evaluation of a frame-based ontology. A formalization-oriented approach. In *Proceedings of MIE2002.*, volume 90, 488–93.

Greiner, R.; Smith, B. A.; and Wilkerson, R. W. 1989. A correction to the algorithm in reiter's theory of diagnosis. *Artif. Intell.* 41(1):79–88.

Haarslev, V., and Möller, R. 2001. RACER system description. In Goré, R.; Leitsch, A.; and Nipkow, T., eds., *IJCAR 2001*, number 2083 in LNAI.

Huang, Z.; van Harmelen, F.; and ten Teije, A. 2005. Reasoning with inconsistent ontologies. In *IJCAI'05*.

McGuinness, D.; Fikes, R.; Rice, J.; and Wilder, S. 2000. The chimaera ontology environment. In *The Seventeenth National Conference on Artificial Intelligence*.

Nebel, B. 1990. Terminological reasoning is inherently intractable. *AI* 43:235–249.

Noy, N., and Musen, M. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*. AAAI Press.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

Schaerf, M., and Cadoli, M. 1995. Tractable reasoning via approximation. *Artificial Intelligence* 74:249–310.

Schlobach, S., and Cornet, R. 2003. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of IJCAI'03*. Morgan Kaufmann.

Schlobach, S. 2005. Semantic clarification by pinpointing. In *Proceedings of ESWC'05*.