Debugging and Semantic Clarification by Pinpointing

Stefan Schlobach

Department of Computer Science,
Vrije Universiteit Amsterdam, The Netherlands
schlobac@few.vu.nl

Abstract. Ontologies are the backbone of the Semantic Web as they allow one to share vocabulary in a semantically sound way. For ontologies, specified in OWL or a related web ontology language, Description Logic reasoner can often detect logical contradictions. Unfortunately, there are two drawbacks: they lack in support for debugging incoherence in ontologies, and they can only be applied to reasonably expressive ontologies (containing at least some sort of negation).

In this paper, we attempt to close these gaps using a technique called *pinpointing*. In pinpointing we identify minimal sets of axioms which need to be removed or ignored to turn an ontology coherent. We then show how pinpointing can be used for *debugging* of web ontologies in two typical cases. More unusual is the application of pinpointing in the *semantic clarification* of underspecified web ontologies which we experimentally evaluate on a number of well-known web-ontologies. Our findings are encouraging: even though semantic ambiguity remains an issue, we show that pinpointing can be useful for debugging, and that it can significantly improve the quality of our semantic enrichment in a fully automatic way.

1 Introduction

Ontologies play a crucial role in the Semantic Web (SW), as they allow "intelligent agents" to share information in a semantically unambiguous way, and to reuse domain knowledge (possibly created by external sources). However, this makes SW technology highly dependent of the quality, and, in particular, of the correctness of the applied ontology. Two general strategies for quality assurance are predominant, one based on developing more and more sophisticated ontology modeling tools, the second one based on logical reasoning. In this paper we will focus on the latter. With the advent of expressive ontology languages such as OWL and its close relation to Description Logics (DL), non-trivial implicit information, such as the `is-a` hierarchy of classes, can often be made explicit by logical reasoners. More crucially, however, state-of-the art DL reasoners can efficiently detect incoherences even in very large ontologies. The practical problem remains what to do in case an ontology has been detected to be incoherent.

Suppose, for example, that an academic researcher wants to make information about his work available in OWL on the Internet. In the spirit of the Semantic

Web, he refers to an existing external ontology about research environments using the ontology `cerif` at [5]. Studying it with an available ontology browser, he finds that the available classes are very useful and that the hierarchy corresponds to his intentions. Unfortunately, the connected logical reasoner tells him that both the concepts *Faculty* and *Institute* are unsatisfiable. What is he supposed to do? Things are even more difficult when our researcher tries to use two or more different ontologies with multiple ownership. As an example, assume that you want to use both the `SUMO` and the `CYC` upper ontologies in a single document. Unfortunately, this attempt leads to over 1000 unsatisfiable concepts.

A second obstacle is that ontologies are often specified in a very limited way. Even though the OWL representation language offers high expressiveness most state-of-the-art online ontologies use fragments of minimal expressiveness.[1] The most prominent example is when the modeling ignores disjointness information of atomic classes, as this renders most logical method for quality assurance useless.[2] This means that, in order to re-introduce logical methods for quality control, we need to semantically enrich the ontology first.

To summarize, to ensure the quality of web-ontologies, there are two major problems to be solved: We call **semantic clarification** the process of automatically enriching ontologies by appropriate disjointness statements which reintroduces the logical functionality in the process of ontological quality control. Secondly, even though incoherences in ontologies might be detected easily, there is usually little support for the identification and elimination of the modeling errors, a problem we call **debugging**. Debugging is a non-trivial process as it is not unusual to deal with over 700 unsatisfiable classes in real applications.

This paper intends to give a *qualitative analysis* over the potential of non-standard reasoning techniques for debugging and semantic clarification. For this purpose, we propose a method called *pinpointing*. Pinpointing is a pragmatic and rather simple technique based on debugging methods we introduced in [16] and boils down to ignoring those ontological axiom which are most likely to be responsible for the incoherence. Before plunging into a more theoretical study of this simple type of *reasoning with inconsistency* we decided to empirically study the practical effect of pinpointing in the Semantic Web. For this purpose we discuss two applications of pinpointing for debugging of ontologies, as introduced in the two previously described scenarios. First, there is the most obvious case of published ontologies which are originally incoherent, and secondly, we consider incoherence as a result of the merging of two or more ontologies. To assess the application of pinpointing for semantic clarification we undertook a number of experiments. We automatically create disjointness statements for "underspecified" ontologies by assuming that all the direct siblings in a well-defined `is-a`

[1] Consider that not even 10% of the ontologies in [5] contain negation.

[2] Inconsistencies in an ontology can be caused by erroneous use of a variety of different language constructs, such as cardinality or role restrictions. The most common reason is disjointness of concepts, though, and we will focus on this class of problem in this paper. None of the described techniques, however, depends on, or is restricted to this particular kind of inconsistencies.

hierarchy should be disjoint. In practice, this assumption is often too strong, and introducing these disjointness statements usually results in a hugely incoherent ontology. We show that, by applying our pinpointing strategy, we can often reduce the number of erroneous disjointness statements significantly.

The main contribution of this paper is threefold: we combine previously known debugging techniques into a framework called pinpointing, which we present and evaluate in the Semantic Web context. Moreover, we show that the application of pinpointing can significantly improve the quality of semantic clarification, a process which in itself is useful for quality assurance of ontologies.

The remainder of this paper is organized as follows: after a brief overview of related work, we give the formal notion of pinpointing in Section 3. In Section 4 we describe three typical cases of pinpointing for debugging of web ontologies. Finally, Section 5 describes our experiments on Semantic Clarification.

2 Related Work

The building of ontologies has been discussed extensively in the literature, and many links can be found on the W3C website [18] about modeling methodology and ontology languages. For Description Logics the handbook [1] is an excellent reference. Explanation has been an issue in the DL community for several years, but most papers, such as [3], deal with subsumption. The number of papers dealing with theoretical studies of reasoning with inconsistency is enormous (just to mention [9, 15]). Our approach in this paper, however, is more humble, as we only investigate the effect of a simple reasoning strategy in an empirical way.

From a more practical point of view, closest to our work are the Chimaera and PROMPT tools ([10] and [12]), which provide support for the merging, analysis and diagnosis of a knowledge base but not, to our knowledge, for debugging. The techniques, that we use to calculate our minimal incoherence preserving sub-TBoxes (MIPSs), however, are similar to those introduced in [2], where the authors use Boolean minimization to calculate minimal inconsistent ABoxes to construct an implicit minimal model for defaults. Finally, our work on semantic clarification is based on the ideas of [4].

3 Pinpointing in Web Ontologies

This paper is about debugging incoherent ontologies in the Semantic Web, i.e., about detection and elimination of logical contradictions in machine-readable formalisations of the shared vocabulary of a particular application domain. We first explain our use of the term ontology and formally define what we mean by incoherence, before introducing our debugging methodology.

3.1 Ontologies in the Semantic Web

In recent years, several semantic web languages have been developed to support the integration of ontologies, and OWL has now been accepted as the recommendation of the W3C as the standard web ontology language [18]. It builds on the XML surface syntax, XML Schema datatypes, the RDF datamodel and RDFS semantics for hierarchies of classes and properties, and it adds more vocabulary to build complex classes and properties. Among others it provides relations between classes, cardinality, enumeration of individuals and Boolean operators. An OWL ontology consists mainly of axioms and facts, where the latter describe properties of individuals and where the former associate classes and properties with possibly complex information. Examples for OWL axioms in the abstract syntax are *Class(Mollusk partial Invertebrate)* and *disjointWith(Mollusk Worm Arthropod)* which states that mollusks are invertebrates but not worms or arthropods. In this paper, we will focus on axioms, and other language constructs available in OWL. OWL has a number of predecessor languages, such as DAML, OIL and DAML-OIL, and many ontologies currently available are simply defined in RDF or RDFS [8]. For the purpose of this paper, we make no distinction between the different types of ontologies and refer to any online collection of axioms in one of the above formalisms as a *web ontology.* On top of RDF, OWL is also rooted in other frameworks such as frames and Description Logics (DL). The connection to DL is useful, as it provides model-theoretic semantics with formally defined reasoning, for which there are several highly optimized and efficient reasoners.

Description Logic Reasoning for Ontologies. We shall not give a formal introduction to Description Logics here, but point to Chapter 2 of [1]. Briefly, DLs are set description languages with concepts, interpreted as subsets of a domain, and roles, interpreted as binary relations. In this paper, we will also use the terms concepts (roles) and classes (properties). In a terminological component $\mathcal{T}$ (called TBox), the interpretation of concepts can be restricted to the *models* of $\mathcal{T}$ by defining *axioms* of the form $C\dot{\sqsubseteq}D$, where C and D are concepts.[3] Based on this formal model-theoretic 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. Other reasoning services include *subsumption* of two concepts (a subset relation w.r.t. all models of $\mathcal{T}$). Subsumption and incoherence are standard reasoning services available in all DL reasoners, such as FaCT [7] and RACER [6]. Although a DL reasoner can classify an ontology and check for the existence of unsatisfiable concepts efficiently, they offer little support for the detection and elimination of errors, i.e., for debugging.

In [16], we proposed a first step to close this gap by introducing a method to explain the incoherence using the notions MUPS, MIPS and cores, which we

[3] Although the definitions and methods in this paper are quite general, the algorithms we implemented and applied in the experiments are restricted to unfoldable $\mathcal{ALC}$ TBoxes. $\mathcal{ALC}$ is a simple yet relatively expressive DL with conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), negation ($\neg C$) and universal ($\forall r.C$) and existential quantification ($\exists r.C$). 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 [11]. Overall, we will not consider assertional components in this paper.

will recall in the following section. But this is not sufficient for debugging, as we will have to repair an incoherent ontology before being able to use it. In Section 3.3 we propose a strategy for fixing the incoherence by pinpointing.

3.2 Explaining Logical Incoherences

In this section, we study ways of *explaining incoherences* in DL terminologies. The idea is to simplify a terminology $\mathcal{T}$ in order to reduce the available information until only the cause of the incoherence remains. More concretely, we exclude axioms that are irrelevant to the incoherence.

To debug an incoherent terminology, we have to identify and eliminate debugging-relevant axioms, where an axiom is *relevant* if a contradictory TBox becomes coherent once the axiom is removed or, at least, a particular, previously unsatisfiable concept becomes satisfiable. Consider the following (incoherent) TBox $\mathcal{T}_1$, where A, B and C are primitive and $A_1, \ldots, A_7$ defined concept names:

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

The set of unsatisfiable concept names as returned by a complete DL reasoner is $\{A_1, A_3, A_6, A_7\}$. Although this is still of manageable size, it hides crucial information, e.g., that unsatisfiability of A_1 depends on unsatisfiability of A_3, which is incoherent because of the contradiction between A_4 and A_5. We will use this example to explain our explanation methods.

Unsatisfiability-preserving sub-TBoxes of a TBox $\mathcal{T}$ and an unsatisfiable concept A are subsets of $\mathcal{T}$ in which A is unsatisfiable. In general, there are several of these sub-TBoxes: and we select the minimal ones, i.e., those containing only axioms that are necessary to preserve unsatisfiability.

Formally, let A be a concept which 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 $\mathcal{T}'' \subset \mathcal{T}'$. We will abbreviate the set of MUPS of $\mathcal{T}$ and A by $mups(\mathcal{T}, A)$. MUPS for our example TBox $\mathcal{T}_1$ and its unsatisfiable concepts are:

$$\begin{aligned}
mups(\mathcal{T}_1, A_1) &= \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}\\
mups(\mathcal{T}_1, A_3) &= \{\{ax_3, ax_4, ax_5\}\}\\
mups(\mathcal{T}_1, A_6) &= \{\{ax_1, ax_2, ax_4, ax_6\},\\
&\qquad \{ax_1, ax_3, ax_4, ax_5, ax_6\}\}\\
mups(\mathcal{T}_1, A_7) &= \{\{ax_4, ax_7\}\}
\end{aligned}$$

MUPS are useful for relating unsatisfiability to sets of axioms but are also the basic ingredients for the calculation of *Minimal Incoherence Preserving Sub-terminologies*, which are the smallest subsets of an original TBox preserving unsatisfiability of at least one atomic concept.

Formally, 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. We will abbreviate the set of MIPSs of $\mathcal{T}$ by $mips(\mathcal{T})$. For our example terminology $\mathcal{T}_1$ we get three MIPSs $mips(\mathcal{T}_1) = \{\{ax_1, ax_2\},\{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$. It can easily be checked that each of the three incoherent TBoxes in $mips(\mathcal{T}_1)$ is indeed a MIPS since taking away a single axiom renders each of the three coherent. The first one signifies, for example, that the first two axioms are already contradictory without reference to any other axiom, which suggests a modeling error already in these two axioms.

Minimal incoherence-preserving sub-TBoxes identify the smallest sets of TBox axioms that cause the original TBox to be incoherent. In terminologies such as `DICE`, which are created through migration from other representation formalisms, there are several such sub-TBoxes, each corresponding to a particular contradictory terminology. *Cores* are now sets of axioms occurring in several of these incoherent TBoxes. The more MIPSs such a core belongs to, the more likely it is that axioms are the cause of contradictions.

Formally, a non-empty intersection of n different MIPSs in $mips(\mathcal{T})$ (with $n \geq 1$) is called a *MIPS-core of arity* n (or n*-ary core*) for $\mathcal{T}$. Every set containing precisely one MIPS is, at least, a 1-ary core. The most interesting cores of a TBox, $\mathcal{T}$, are those with axioms that are present in as many MIPSs of $\mathcal{T}$ as possible, i.e., having maximal arity. On the other hand, the size of a core is also significant, as a larger core indicates that clusters of axioms cause contradictions in combination only. In our example, axiom ax_4 occurs both in $\{ax_3, ax_4, ax_5\}$ and $\{ax_4, ax_7\}$, which makes $\{ax_4\}$ a core of arity 2 and size 1 for $\mathcal{T}_1$, which is the core of maximal arity in this example.

There is no unique way of deciding which axioms in the MIPS are the most relevant for the incoherence. Our approach is pragmatic: we assume that an axiom is more likely to be erroneous the more often it occurs in the set of MIPS. Therefore, we look for cores with maximal arity.

The general definitions of MUPS, MIPS and cores do not depend on a particular ontology language and can easily be extended to include facts about individuals. However, the algorithms we implemented are restricted to unfoldable $\mathcal{ALC}$ terminologies. It was shown in [16] that the problem of calculating MIPS for an $\mathcal{ALC}$ terminology is in PSPACE. Nevertheless, in our experience calculating all the MIPS was practically feasible in all but a few cases.[4] This is due to the relatively simple structure of the ontologies we considered. For more complex cases, approximative methods need to be investigated. Similarly, checking elements of $mips(\mathcal{T})$ for cores of maximal arity requires exponentially many checks in the size of $mips(\mathcal{T})$. For efficiency reasons, we therefore currently only check for cores of size 1.

[4] The overall run-times for the experiments described in Section 4 and 5 were minutes rather than hours even for the most complex ontologies.

3.3 A Strategy for Fixing Incoherences

MIPSs, MUPS and cores do not offer an immediate recipe for fixing an incoherent web ontology. For web ontologies we propose a strategy which iteratively calculates the cores (of size 1) of maximal arity, repeating the process on the remaining MIPSs. Moreover, the algorithms for MIPS and MUPS have been implemented for terminological debugging of $\mathcal{ALC}$ terminologies. To apply them to web ontologies we need some preprocessing. Given a web ontology $\mathcal{O}$ we apply the following steps:

1. Remove the ABox, as well as property statements and axioms where the left-hand side is non-atomic.
2. Replace equivalence statements by implications
3. Collect all implications $C \sqsubseteq D_1, \ldots, C \sqsubseteq D_n$ in a single conjunction $C \sqsubseteq D_1 \sqcap \cdots \sqcap D_n$.
4. Call the resulting terminology $\mathcal{T}$ (not necessarily unfoldable)

The strategy for automatically fixing the incoherences by pinpointing is then as follows; calculate:

1. the set $Unsat(\mathcal{T})$ of unsatisfiable concept-names in $\mathcal{T}$ using RACER;
2. the MUPS $mups(\mathcal{T}, CN)$ for all concept-names $CN \in Unsat(\mathcal{T})$;
3. the MIPSs $mips(\mathcal{T})$ for $\mathcal{T}$ from the MUPS;
4. let $M := mips(\mathcal{T})$, $P(\mathcal{O}) = \varnothing$. Now calculate while $M \neq \varnothing$:
   (a) the core $\{ax\}$ of M of size 1 with maximal arity, and add it to $P(\mathcal{O})$;
   (b) remove from M the mips containing ax.
5. $P(\mathcal{O})$ will be called the *pinpoint of* $\mathcal{O}$.
6. Finally, remove $P(\mathcal{O})$ from $\mathcal{T}$.

The pinpoint of the ontology $\mathcal{O}$ is a set of axioms in the preprocessed version. Every axiom corresponds precisely to a concept-name (because of step 3) or is a disjointness statement. By the pinpoint of an ontology, we will therefore refer both to sets of axioms as well as to sets of concept and disjointness statements. Note that for debugging and semantic clarification, we often focus on these pinpoints. This is because there is usually only a handful of those as compared to hundreds of MIPSs, which can be quite complicated. However, in practice, MIPSs will always have to be consulted if one wants to understand the underlying reasons for incoherence of an ontology.

Fixing an ontology by pinpointing offers a simple solution to the incoherence problem since by adjusting or removing the information about the pinpoints we can guarantee to the restoration of logical coherence with (almost) minimal intrusion in the ontology. Pinpoints correspond to hitting-sets [14] for the MIPS, but they are not necessarily minimal. Take, as example, a set $M = \{\{ax_1, ax_2\}, \{ax_3, ax_4\}, \{ax_5, ax_1\}, \{ax_5, ax_3\}\}$ of MIPS, with $\{ax_5, ax_1, ax_3\}$ as possible pinpoint, even though $\{ax_1, ax_3\}$ is a miminal covering set for M. On the other hand, it has to be remarked that calculating minimal hitting-sets from the set of MIPS is NP-COMPLETE, as compared to linear time to calculate pinpoints from the MIPS.

We evaluated our strategy of debugging of web ontologies by pinpointing on a number of applications. The goal of our experiments was twofold, first to assess some case studies on more typical applications of debugging, but also to put forward a more unconventional approach to resolving semantic ambiguity that we claim can be supported by our proposed methods.

4 Debugging Web-Ontologies by Pinpointing

Pinpointing was initially developed for debugging of the medical terminology `DICE`. To evaluate whether the method use useful for and scales to web ontologies we looked at two case studies: first, importing an unsatisfiable ontology, and, secondly, merging several ontologies.

4.1 Import an *Institute* and Ignore the *Faculty*:

Remember our introductory example from the academic, who plans to use the `cerif` ontology, which defines two concepts in unsatisfiable ways. This example demonstrates the simplest application of debugging by pinpointing, namely when a user wants to import an incoherent ontology. Our suggested strategy for debugging is as follows: whenever we detect an incoherence in an ontology we trace down the most likely source of the logical contradiction by calculating the pinpoint for it. For the `cerif` ontology there are two MIPSs with a single concept *Faculty* occurring in both, i.e., the pinpoint contains only *Faculty*. This tells us that the source of the logical incorrectness of the definition of *Institute* is probably the incorrect definition of *Faculty*. At least theoretically, this would allow the user to "rescue" the *Institute* class, by overwriting or simply ignoring the *Faculty* concept.

Although the `cerif` is a real-world example, incoherent ontologies are rarely published on-line. In contrast, debugging at creation or migration time is quite typical. Let us describe how pinpointing was used when the `DICE` terminology was migrated from frames to a Description Logics representation.

4.2 Is a Physical Quantity Temporal or Mathematical?

The previous example illustrated the use of debugging by pinpointing of a single ontologies. Most people, however, will use several ontologies and in this case, might be even more likely to encounter incoherence. In the Introduction we mentioned the example of using both the `SUMO` and `CYC` upper ontologies. As they are topic-related, and as `CYC` provides disjointness statements, there is indeed a high number of unsatisfiable concepts. Let us define our observation in more detail, starting with a description of the ontologies.

– `SUMO`: the *Suggested Upper Merged Ontology* is an upper level ontology suggested by the IEEE Suggested Upper Ontology Working group. It was created by the Teknowledge Corporation and was made publicly available at `http://ontology.teknowledge.com` [17].

– CYC: OpenCyc is the open source version of the Cyc technology, “the world’s largest and most complete general knowledge base.” CYC includes about 6,000 concepts – an upper ontology for all of human consensus reality – and 60,000 assertions about the 6,000 concepts, interrelating them, constraining them, in effect (partially) defining them. http://www.opencyc.org. [13]

In our experiments we removed the unique name-spaces from both SUMO and CYC and syntactically merged the two ontologies. We then used RACER to check for coherence, which resulted in an un-ordered list of 1093 unsatisfiable concepts. Initially this seems to be a discouraging result. However, the pinpoint of the joint ontology contains only 4 concept names *Event*, *Product*, *Entity* and *Tuple*. This is surprising as it means that we can ensure coherence of the merged ontology by ignoring (or fixing) the axioms defining *Event*, *Product*, *Entity* and *Tuple*.

Let us try to get a better intuition for the problem. Each of the four concepts defined by the axioms in the pinpoints occur in contradictions in a variety of different combinations of other concepts. Let us look at one of the MIPSs, here in form of the derivation of one of the contradictions, namely of the concept *Physicalquantity*. Here, the upward derivation stems from the SUMO hierarchy and the downward derivation from CYC, as a *Physicalquantity* is defined as a *quantity* in SUMO and a *Scalarinterval* in CYC. A contradiction occurs as the class *MathematicalOrComputationalThing* is defined to be disjoint from *Temporalthing* in CYC. Interestingly enough it is definition of the concept *Entity* as a *Temporalthing* that is to be ignored by our strategy, which is what constitutes one of the “philosophical” difference between the SUMO and CYC ontologies.

$$\begin{array}{l}
\textit{Temporalthing}\\
\quad\quad \textit{Somethingexisting}\\
\quad\quad\quad\quad \underline{\textit{Entity}}\\
\quad\quad\quad\quad\quad\quad \textit{Abstract}\\
\quad\quad\quad\quad\quad\quad\quad\quad \textit{Quantity}\\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \boxed{\textit{Physicalquantity}}\\
\quad\quad\quad\quad\quad\quad\quad\quad \textit{Scalarinterval}\\
\quad\quad\quad\quad\quad\quad \textit{NTupleinterval}\\
\quad\quad\quad\quad \textit{Tuple}\\
\quad\quad\quad \textit{MathematicalObject}\\
\quad\quad \textit{Mathematicalthing}\\
\textit{MathematicalOrComputationalThing}
\end{array}$$

For the naive user of the two ontologies, the practical benefit of pinpointing is immediate. Discarding the axioms defining the four elements of the pinpoint will directly render the ontology coherent and all the remaining 1089 previously unsatisfiable concepts satisfiable, and therefore usable again. This restores most[5] of the useful reasoning facilities which users have learned to expect from their ontology development tools.

[5] Removing the axioms obviously has side effects. In the case of the concept *Entity* this implies that none of the descendants of *Entity* can be classified as a TEMPORALTHING any more. In this sense a certain CYC view of the world predominates.

5 Semantic Clarification by Pinpointing

To improve the quality of the `DICE` terminology, the developers migrated a frame-based to a DL-based representation. For this purpose, they made a number of very strong assumptions, as they had to decide on the semantic interpretations of operators such as slot-fillers or super-classes. One of the main issues was to make disjointness between classes explicit. In the frame-based version of `DICE`, it is impossible, for example, to state that nothing can be both a *Liver* and a *Kidney*, i.e., that the classes are disjoint. However, a controlled medical terminology should prohibit the definition of a particular organ as a subclass of $Liver \sqcap Kidney$. In [4] Cornet and Abu-Hannah discuss a number of assumptions they base their migration on. The most interesting for this paper is the:

> **Strong Disjointness Assumption (SDA):** In a well-modeled terminology the direct siblings, i.e. children of a common parent in the subsumption hierarchy should be disjoint.

In a Semantic Web application, we face a similar dilemma as in the migration of `DICE`. Many publicly available ontologies have been created by migration from frame-based representations, but usually, relatively weak a priori assumptions were made in the migration process (such as not including any disjointness statements). In this paper, we propose a more rigid migration strategy that should help the user end up with a version of his/her ontology where at least some basic disjointness relation between atomic concepts has been established. This is achieved by applying the Strong Disjointness Assumption and using a mechanism to deal with exceptions. In our case, this mechanism is based on pinpointing. Let us describe the general strategy.

5.1 The Strategy for Semantic Clarification

Given a semantically weakly specified ontology $\mathcal{O}$ (without disjointness statements) we, first, add all possible disjointness statements according to the SDA to $\mathcal{O}$ before debugging the ensuing incoherences by pinpointing. More formally this consists of the following steps:

1. Use RACER to classify $\mathcal{O}$.
2. Let $D = \{\{C_1^1, \ldots, C_{n_1}^1\}, \ldots, \{C_1^m, \ldots, C_{n_m}^m\}\}$ be the set of all sets of concept names which have at least one common parent in the subsumption hierarchy.
3. The set $sug\_disj(\mathcal{O}) = \{disjoint(C_1^1, \ldots, C_{n_1}^1), \ldots, disjoint(C_1^m, \ldots, C_{n_m}^m)\}$ contains all the disjointness statements suggested given the SDA.
4. Add $sug\_disj(\mathcal{O})$ to $\mathcal{O}$ to create the possibly incoherent $\mathcal{O}^* = \mathcal{O} \cup sug\_disj(\mathcal{O})$.
5. Calculate the pinpoint $P(\mathcal{O}^*)$ for $\mathcal{O}^* = \mathcal{O} \cup sug\_disj(\mathcal{O})$
6. Remove the disjointness statements $D \in P(\mathcal{O}^*)$ from $\mathcal{O}^*$ to make it coherent.

5.2 Experiments

To evaluate clarification by pinpointing we applied our strategy on ontologies, first, to assess the quality of the added disjointness statements given our strong disjointness assumption, and secondly, to study how much pinpointing can improve on these results. We do this in three steps, first, we evaluate the relation of the size of an ontology with the damage inflicted on the ontology by adding disjointness statements. Secondly, we look at the quality of the disjointness statements themselves. Finally, we check whether more information improves the semantic clarification. First, however, we discuss the data used for the experiments.

The Data (Web Ontologies): The problem with an evaluation such as ours is that it requires domain knowledge to evaluate the quality of the disjointness statements. Therefore, we focused on general knowledge ontologies and on ontologies describing domains where we have some expertise (such as soccer). We used the following ontologies:

- `MGED`: provides standard terms for the annotation of microarray experiments in order to enable structured queries on those experiments;
- `UNSPSC`: a translated version of the *Universal Standard Products and Services Classification Code* which provides an open, global multi-sector standard for efficient, accurate classification of products and services;
- `soccer`: "concepts that are specific to soccer: players, rules, field, supporters, actions, etc. Used to annotate videos." [5]
- `SUMO`: (as described above);
- `MILO`: a midlevel ontology that acts as a bridge between the high-level abstractions of the `SUMO` and the low-level detail of the domain ontologies;
- `Eco`: an ontology describing properties specific to the economy;
- `Trans`: an ontology of terms about transportation-related information;
- `Gov`: an ontology of government concepts;
- `Geo`: an ontology of geography.

The last 6 ontologies were all made available by the Teknowledge Corporation and more details can be found at [17]. Most information for the last 4 ontologies is taken from the CIA World Fact Book (2002), but many other sources are used.

Question 1: Is the Size of an Ontology Relevant for Semantic Clarification? In the first set of experiments, we wanted to study what role the structure and size of an ontology plays when adding disjointness statements for clarification. For this purpose, we added disjointness statements to the ontologies described above and calculated the set of unsatisfiable concepts and the MIPSs. We take a high number of unsatisfiable concepts and many MIPSs as an indicator that the Strong Disjointness Assumption is inadequate for the given ontology. Table 1 gives an overview of the number of implications (the only axioms we consider), the number of disjointness statements created, the number of unsatisfiable concepts in the ontology with the disjointness statements, and the number of MIPSs.

Table 1. Adding disjunctions to web ontologies

$\mathcal{O} =$	`MGED`	`UNSPSC`	`soccer`	`SUMO`	`MILO`	`Eco`	`Trans`	`Gov`	`Geo`
#axioms	370	9795	194	669	1764	409	455	50	417
#disj	34	1463	40	169	342	60	89	12	82
#Unsat($\mathcal{O}^*$)	72	0	0	175	46	213	145	0	11
#*mips*($\mathcal{O}^*$)	42	0	0	149	59	154	189	0	22

Note that we have small ontologies which become incoherent and that the big ontology UNSPSC remains coherent even after adding over 1000 disjointness statements. It shows that the size of an ontology is not the main factor in semantic clarification. There is not even a unique pattern for the 4 specialized ontologies provided by Teknowledge: whereas the geography ontology `Geo` has only very few (11) unsatisfiable concepts, there are now 89 unsatisfiable concepts in the transportation ontology `Trans` which is of similar size, although one would expect comparable modeling. It is clear that a more careful analysis is needed, taking a closer look at the created disjointness statements.

Question 2: How Useful are Disjointness Statements? In the next step of the experiments, each of the created disjointness statements was evaluated by a human assessor as true or false. In some cases, the evaluators did not have enough domain knowledge so the numbers do not always add up. Unfortunately, we could not further experiment with `UNSPSC` and `MILO` because of their size and we did not consider `MGED` because of our lack of expert knowledge. Table 2 summarizes the results for the evaluation of the quality of the created disjointness statements and our pinpointing based debugging method.

Remember that the soccer ontology `soccer` and both `Gov` and `GEO` were coherent when adding the disjointness statements so that, obviously, no MIPS could be found. Astonishingly, there is again a big discrepancy between the `Trans` and the `Geo` ontology, which does not even disappear after removing the pinpointed disjointness statements. It is difficult to see the reasons for this odd behavior of two ontologies that are, in principle, of a very similar structure, size and pedigree, but we suppose that it is due to differences in modeling. We look now at some examples to get a better understanding of what might go wrong.

Error Analysis: The `soccer` ontology adds a level of structure to the class of players by separating goalkeepers, other players and substitutes. This, however, is strange modeling practice, as goalkeepers can be substitutes. The disjointness statement *(disjoint Goalkeeper OtherPlayer Substitute)* which would be added by our clarification strategy would therefore be erroneous. As we do not have more information about goalkeepers and substitutes, this error cannot be found using our pinpointing strategy.

In the `Geo` ontology volcanoes and upland areas are both classified as landforms, which suggests the addition of *(disjoint ... UplandArea Volcano)* (we will

Table 2. Evaluating the Quality of the Disjointness Statements

$\mathcal{O} =$	`soccer`	`SUMO`	`Eco`	`Trans`	`Gov`	`Geo`
#disj	40	169	60	89	14	82
#false disj	5	63	14	49	7	19
#unknown	0	8	0	17	0	8
#correct disj	35	98	46	23	7	55
#correct in %	87%	60%	76%	31%	50%	74%
#improved by removing $P(\mathcal{O})$	0	25	5	15	0	5
#correct after improving in %	87%	76%	85%	52%	50%	81%

refer to this statement as DISS) according to our clarification strategy. Here, pinpointing can help, as we now have an unsatisfiable concept *VolcanicMountain* with two MIPSs {*Mountain, Volcano, VolcanicMountain, DISS*} {*UplandArea, Mountain, VolcanicMountain, DISS*}, and a subsequent pinpoint DISS.[6] What this suggests is the following: pinpoints can be useful when there is enough specialized knowledge to make the exceptions to the disjointness statements explicit. In the above case, this is the knowledge about a volcanic mountain, which is both an Upland and a Volcano, helps us find the error.

The observation that more specialized knowledge helps to improve the quality of the pinpointing leads us to a final round of experiments on the `SUMO` ontology.

Question 3: Does More Knowledge Help Debugging? To assess the claim that additional information can make pinpointing more successful, we combined `SUMO` with the more specialized ontologies `MILO, Eco, Trans, Gov` and `Geo` from the Teknowledge family. Remember that with pinpointing, we were able to improve the quality of the added disjointness statements by 16% in our previous experiment, but 38 statements remained erroneous. For the last experiment, we added to `SUMO` the 169 disjointness statements following from the Strong Disjointness Assumption and removed from it the 25 occurring in the pinpoint. The resulting ontology is coherent. Adding `MILO` to this ontology, however, creates new incoherences, and we find a further 15 erroneous disjointness statements. Adding the 4 specialized ontologies `Trans,Eco, Geo` and `Gov` also helps detecting 2 more false disjunctions, which leaves the new `SUMO` (with pinpoints removed) with 86% correct disjointness statements (up from 60%).

Erroneous Disjointness Statements? Even after pinpointing we are left with 14% incorrect disjointness statements in the above experiment. This sounds dangerous, but has to be put in the perspective of the application. Remem-

[6] If an incoherent terminology is the result of a semantic clarification process, it is part of the assumption that some of the new, artificially created, statements are likely to be incorrect. Therefore, whenever we have a choice, (as in this example) we include the disjointness statement into the pinpoint.

ber, that disjointness statements are only interesting for reasoning to establish coherence, and that they do not influence the subsumption hierarchy, given that the ontology is indeed coherent. What remains is the danger that logical contradictions occur when creating new objects referring to semantically enriched class. But then, again, pinpointing offers a simple solution: if it is indeed the disjointness axiom that causes the contradiction it alone will constitute the pinpoint, and will automatically be removed to render the new object satisfiable.

6 Conclusion and Further Work

With the arrival of more expressive ontology languages in the Semantic Web community, incoherences increasingly become a problem that can seriously hamper the construction and application of web ontologies. In this paper we presented a strategy for automatically identifying and fixing incoherences that is based on first explaining their causes and, secondly, choosing (and eliminating) axioms that most frequently participate in the underlying logical contradictions.

We discussed our pinpointing strategy with respect to standard debugging of incoherences. Then, we showed how pinpointing can help the semantic clarification of underspecified ontologies. For the first case-studies, we checked a number of web ontologies for coherence and explored what happens when the two upper-level ontologies `CYC` and `SUMO` are merged. The most interesting finding was that, although there was a very high number of unsatisfiable concepts, the pinpoint only consisted of 4 elements. We suggest that studying these concepts in more detail can help in clarifying the alternative modeling approaches.

The second case-study was an attempt to clarify the semantics of web ontologies for which no disjointness (or negation) of classes is specified. In this case, incoherence is impossible, but we can automatically add disjointness statements if we assume that direct siblings in a hierarchy should be disjoint. We evaluated this assumption on a number of web ontologies by first including these disjointness statements and, subsequently, applying our pinpointing strategy. The empirical and qualitative results showed that too many disjointness statements remained because not all exceptions to our assumption were covered. However, we already outlined a way out of this dilemma, which is to extend general ontologies with more specific ontologies to better identify exceptions and therefore to exclude the erroneous disjointness statements. Conceptually, it should be a simple extension to add facts, but the additional computational complexity might render the approach infeasible in practice.

Finally, it remains to study our pinpointing approach to reasoning with incoherent ontologies from a theoretical perspective. Surely enough ignoring the ontological axiom from the cores renders the ontology coherent, but nothing can be said so far about the quality (such as maximality or meaningfulness) of the resulting ontology.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. Technical Report RR-93-20, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 1993.
3. A. Borgida, E. Franconi, and I. Horrocks. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence*, pages 209–213, 2000.
4. R. Cornet and A. Abu-Hanna. Evaluation of a frame-based ontology. A formalization-oriented approach. In *Proceedings of MIE2002, Studies in Health Technology & Information*, volume 90, pages 488–93, 2002.
5. DAML ontology library. URL. http://www.daml.org/ontologies/.
6. V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, number 2083 in LNAI, 2001.
7. I. Horrocks. The FaCT system. In H. de Swart, editor, *Tableaux'98*, number 1397 in LNAI, pages 307–312, 1998.
8. I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
9. Pierre Marquis Jrme Lang. Removing inconsistencies in assumption-based theories through knowledge-gathering actions. *Studia Logica*, 67(2):179–214, 2001.
10. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera Ontology Environment. In *The Seventeenth National Conference on Artificial Intelligence*, 2000.
11. B. Nebel. Terminological reasoning is inherently intractable. *AI*, 43:235–249, 1990.
12. N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000.
13. http://www.opencyc.org/.
14. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
15. M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artif. Intell.*, 74(2):249–310, 1995.
16. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence, IJCAI'03*. Morgan Kaufmann, 2003.
17. Sumo (Suggested Upper Merged Ontology). URL, as visited on April 16, 2004. http://ontology.teknowledge.com/.
18. Web-ontology (WebOnt) working group of the W3C. URL, as of April 16, 2004. http://www.w3.org/2001/sw/WebOnt/.