Query strategy for sequential ontology debugging

Kostyantyn Shchekotykhin and Gerhard Friedrich *

Universitaet Klagenfurt
Universitaetsstrasse 65-67
9020 Klagenfurt, Austria
firstname.lastname@ifit.uni-klu.ac.at

Abstract. Debugging is an important prerequisite for the wide-spread application of ontologies, especially in areas that rely upon everyday users to create and maintain knowledge bases, such as the Semantic Web. Most recent approaches use diagnosis methods to identify sources of inconsistency. However, in most debugging cases these methods return many alternative diagnoses, thus placing the burden of fault localization on the user. This paper demonstrates how the target diagnosis can be identified by performing a sequence of observations, that is, by querying an oracle about entailments of the target ontology. We exploit probabilities of typical user errors to formulate information theoretic concepts for query selection. Our evaluation showed that the suggested method reduces the number of required observations compared to myopic strategies.

1 Introduction

The application of semantic systems, including the Semantic Web technology, is largely based on the assumption that the development of ontologies can be accomplished efficiently even by every day users. However, studies in cognitive psychology, like [1], discovered that humans make systematic errors while formulating or interpreting logical descriptions. Results presented in [10, 12] confirmed these observations regarding ontology development. Therefore it is essential to create methods that can identify and correct erroneous ontological definitions. Ontology debugging tools simplify the development of ontologies by localizing a set of axioms that should be modified in order to formulate the intended target ontology.

To debug an ontology a user must specify some requirements such as coherence and/or consistency. Additionally, one can provide test cases [3] which must be fulfilled by the target ontology $\mathcal{O}_t$. A number of ontology diagnosis methods have been developed [13, 6, 3] to pinpoint alternative sets of possibly faulty axioms (called a set of diagnoses). A user has to change at least all of the axioms of one diagnosis in order to satisfy all of the requirements and test cases.

However, the diagnosis methods can return many alternative diagnoses for a given set of test cases and requirements. A sample study of real-world inconsistent ontologies presented in Table 1 shows that even a small number of irreducible sets of axioms that are together inconsistent/incoherent (conflict sets) can be a source of a large number of diagnoses. For instance only 8 conflict sets in the Economy ontology resulted in

* The research project is funded by grants of the Austrian Science Fund (Project V-Know, contract 19996)

	Ontology	Axioms	#C/#P/#I	#CS/min/max	#D/min/max	Domain
1.	Chemical	114	48/20/0	6/5/6	6/1/3	Chemical elements
2.	Sweet-JPL	2579	1537/121/50	8/1/13	13/8/8	Earthscience
3.	University	50	30/12/4	4/3/5	90/3/4	Training
4.	Tambis	596	395/100/0	7/3/9	147/3/7	Biological science
5.	Economy	1781	339/53/482	8/3/4	864/4/9	Mid-level
6.	Transport	1300	445/93/183	9/2/6	1782/6/9	Mid-level

Table 1. Dianosis results for some real-world ontologies presented in [6]. #C/#P/#I are the numbers of concepts, properties, and individuals in an ontology. #CS/min/max are the number of conflict sets, their minimum and maximum cardinality. The same notation is used for diagnoses #D/min/max. These ontologies are available upon request

864 diagnoses. In the case of Transportation ontology the diagnosis method was able to identify 1782 diagnoses. In such situations simple visualization of all alternative changes of the ontology is ineffective.

A possible solution would be to introduce an ordering using some preference criteria. For instance, Kalyanpur et al. [7] suggest measures to rank the axioms of a diagnosis depending on their structure, occurrence in test cases, etc. Only the top ranking diagnoses are then presented to the user. Of course this set of diagnoses will contain the target one only in the case when a faulty ontology, the given requirements and test cases, provide sufficient data to appropriate heuristics. However, in most debugging sessions a user has to provide additional information (e.g. in the form of tests) to identify the target diagnosis.

In this paper we present an approach to acquisition of additional information by generating a sequence of queries, which should be answered by some oracle such as a user, an information extraction system, etc. Our method uses each answer to a query to reduce the set of diagnoses until finally it identifies the target diagnosis. In order to construct queries we exploit the property that different diagnoses imply unequal sets of axioms. Consequently, we can differentiate between diagnoses by asking the oracle if the target ontology should imply an axiom or not. These axioms can be generated by classification and realization services provided in description logic reasoning systems [15, 4]. In particular, the classification process computes a subsumption hierarchy (sometimes also called "inheritance hierarchy" of parents and children) for each concept name mentioned in a TBox. For each individual mentioned in an ABox, realization computes the atomic concepts (or concept names) of which the individual is an instance [15].

In order to generate the most informative query we exploit the fact that some diagnoses are more likely than others because of typical user errors. The probabilities of these errors can be used to estimate the change in entropy of the set of diagnoses if a particular query is answered. We select those queries which minimize the expected entropy, i.e. maximize the information gain. An oracle should answer these queries until a diagnosis is identified whose probability is significantly higher than those of all other diagnoses. This diagnosis is the most likely to be the target one.

We compare our entropy-based method with a greedy approach that selects those queries which try to cut the number of diagnoses in half as well as with a "random" strategy when the algorithm selects queries to be asked completely randomly. The evaluation was performed using the set of ontologies presented in Table 1 and generated

examples. Its results show that on average the suggested entropy-based approach is at least 50% better than the greedy one.

The remainder of the paper is organized as follows: Section 2 presents two introductory examples as well as the basic concepts. The details of the entropy-based query selection method are given in Section 3. Section 4 describes the implementation of the approach and is followed by evaluation results in Section 5. The paper concludes with an overview of related work.

2 Motivating examples and basic concepts

In order to explain the fundamentals of our approach let us introduce two examples.

Example 1 Consider a simple ontology $\mathcal{O}$ with the terminology $\mathcal{T}$:

$$ax_1 : A \sqsubseteq B \qquad ax_2 : B \sqsubseteq C \qquad ax_3 : C \sqsubseteq Q \qquad ax_4 : Q \sqsubseteq R$$

and the background theory $\mathcal{A} : \{A(w), \neg R(w)\}$. Let the user explicitly define that the two assertional axioms should be considered as correct.

The ontology $\mathcal{O}$ is inconsistent and the only irreducible set of axioms (minimal conflict set) that preserves the inconsistency is $CS : \{\langle ax_1, ax_2, ax_3, ax_4\rangle\}$. That is one has to modify or remove the axioms of at least one diagnosis:

$$\mathcal{D}_1 : [ax_1] \quad \mathcal{D}_2 : [ax_2] \quad \mathcal{D}_3 : [ax_3] \quad \mathcal{D}_4 : [ax_4]$$

to restore the consistency of the ontology. However it is unclear, which diagnosis from the set $\mathbf{D} : \{\mathcal{D}_1 \ldots \mathcal{D}_4\}$ corresponds to the target one.

In order to focus on the essentials of our approach we employ the following simplified definition of diagnosis without limiting its generality. A more detailed version can be found in [3].

We allow the user to define a background theory (represented as a set of axioms) which is considered to be correct, a set of logical sentences which must be implied by the target ontology and a set of logical sentences which must *not* be implied by the target ontology. Following the standard definition of the diagnosis [11, 8], we assume that each axiom $ax_j \in \mathcal{D}_i$ is faulty whereas each axiom $ax_k \notin \mathcal{D}_i$ is correct.

Definition 1. *Given a diagnosis problem $\langle \mathcal{O}, B, T^{\models}, T^{\not\models}\rangle$ where $\mathcal{O}$ is an ontology, B a background theory, $T^{\models}$ a set of logical sentences which must be implied by the target ontology $\mathcal{O}_t$, and $T^{\not\models}$ a set of logical sentences which must* not *be implied by $\mathcal{O}_t$.*

A diagnosis is a set of axioms $\mathcal{D} \subseteq \mathcal{O}$ such that the set of axioms $\mathcal{O} \setminus \mathcal{D}$ can be extended by a logical description EX and $(\mathcal{O} \setminus \mathcal{D}) \cup B \cup EX \models t^{\models}$ for all $t^{\models} \in T^{\models}$ and $(\mathcal{O} \setminus \mathcal{D}) \cup B \cup EX \not\models t^{\not\models}$ for all $t^{\not\models} \in T^{\not\models}$.

A diagnosis $\mathcal{D}$ is minimal if there is no proper subset of the faulty axioms $\mathcal{D}' \subset \mathcal{D}$ such that $\mathcal{D}'$ is a diagnosis. The following proposition allows us to characterize diagnoses without the extension EX. The idea is to use the sentences which must be implied to approximate EX.

Corollary 1. *Given a diagnosis problem $\langle \mathcal{O}, B, T^{\models}, T^{\not\models}\rangle$, a set of axioms $\mathcal{D} \subseteq \mathcal{O}$ is a diagnosis iff $(\mathcal{O} \setminus \mathcal{D}) \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \cup \neg t^{\not\models}$ consistent for all $t^{\not\models} \in T^{\not\models}$.*

In the following we assume that a diagnosis always exists under the (reasonable) condition that the background theory together with the axioms in $T^{\models}$ and the negation of axioms in $T^{\not\models}$ are mutually consistent. For the computation of diagnoses the set of conflicts is usually employed.

Definition 2. *Given a diagnosis problem $\langle \mathcal{O}, B, T^{\models}, T^{\not\models} \rangle$, a conflict set $CS \subseteq \mathcal{O}$ is a set of axioms s.t. there is a $t^{\not\models} \in T^{\not\models}$ and $CS \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \cup \neg t^{\not\models}$ is inconsistent.*

A conflict is the part of the ontology that preserves the inconsistency/incoherency. A minimal conflict CS has no proper subset which is a conflict. $\mathcal{D}$ is a (minimal) diagnosis iff $\mathcal{D}$ is a (minimal) hitting set of all (minimal) conflict sets [11].

In order to differentiate between the minimal diagnoses $\{\mathcal{D}_1 \dots \mathcal{D}_4\}$ an oracle can be queried for information about the entailments of the target ontology. For instance, in our example the ontologies $\mathcal{O}_i = \mathcal{O} \setminus \mathcal{D}_i$ have the following entailments $\mathcal{O}_1 : \emptyset$, $\mathcal{O}_2 : \{B(w)\}$, $\mathcal{O}_3 : \{B(w), C(w)\}$, and $\mathcal{O}_4 : \{B(w), C(w), Q(w)\}$ provided by the realization of the ontology. Based on these entailments we can ask the oracle whether the target ontology has to entail $Q(w)$ or not ($\mathcal{O}_t \not\models Q(w)$). If the answer is *yes* (which we model with the boolean value 1), then $Q(w)$ is added to $T^{\models}$ and $\mathcal{D}_4$ is the target diagnosis. All other diagnoses are rejected because $(\mathcal{O} \setminus \mathcal{D}_i) \cup B \cup \{Q(w)\}$ for $i = 1, 2, 3$ is inconsistent. If the answer is *no* (which we model with the boolean value 0), then $Q(w)$ is added to $T^{\not\models}$ and $\mathcal{D}_4$ is rejected as $(\mathcal{O} \setminus \mathcal{D}_4) \cup B \models Q(w)$ (rsp. $(\mathcal{O} \setminus \mathcal{D}_4) \cup B \cup \neg Q(w)$ is inconsistent) and we have to ask the oracle another question.

Property 1. Given a diagnosis problem $\langle \mathcal{O}, B, T^{\models}, T^{\not\models} \rangle$, a set of diagnoses $\mathbf{D}$, and a set of logical sentences X representing the query $\mathcal{O}_t \models X$:

If the oracle gives the answer *1* then every diagnosis $\mathcal{D}_i \in \mathbf{D}$ is a diagnosis for $T^{\models} \cup X$ iff $(\mathcal{O} \setminus \mathcal{D}_i) \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \cup \{X\} \cup \neg t^{\not\models}$ is consistent for all $t^{\not\models} \in T^{\not\models}$.

If the oracle gives the answer *0* then every diagnosis $\mathcal{D}_i \in \mathbf{D}$ is a diagnosis for $T^{\not\models} \cup \{X\}$ iff $(\mathcal{O} \setminus \mathcal{D}_i) \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \cup \neg X$ is consistent.

Note, a set X corresponds to a logical sentence where all elements of X are connected by $\wedge$. This defines the semantic of $\neg X$.

As possible queries we consider sets of entailed concept definitions provided by a classification service and sets of individual assertions provided by realization. In fact, the intention of classification is that a model for a specific application domain can be verified by exploiting the subsumption hierarchy [2].

One can use different methods to select the best query in order to minimize the number of questions asked to the oracle. "Split-in-half" heuristic is one of such methods that prefers queries which remove half of the diagnoses from the set $\mathbf{D}$. To apply this heuristic it is essential to compute the set of diagnoses that can be rejected depending on the query outcome. For a query X the set of diagnoses $\mathbf{D}$ can be partitioned in sets of diagnoses $\mathbf{D}^{\mathbf{X}}$, $\mathbf{D}^{\neg\mathbf{X}}$ and $\mathbf{D}^{\emptyset}$ where

- for each $\mathcal{D}_i \in \mathbf{D}^{\mathbf{X}}$ it holds that $(\mathcal{O} \setminus \mathcal{D}_i) \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \models X$
- for each $\mathcal{D}_i \in \mathbf{D}^{\neg\mathbf{X}}$ it holds that $(\mathcal{O} \setminus \mathcal{D}_i) \cup B \cup \{\bigwedge_{t^{\models} \in T^{\models}} t^{\models}\} \models \neg X$
- $\mathbf{D}^{\emptyset} = \mathbf{D} \setminus (\mathbf{D}^{\mathbf{X}} \cup \mathbf{D}^{\neg\mathbf{X}})$

Given a diagnosis problem we say that the diagnoses in $\mathbf{D}^{\mathbf{X}}$ predict 1 as a result of the query X, diagnoses in $\mathbf{D}^{\neg\mathbf{X}}$ predict 0, and diagnoses in $\mathbf{D}^{\emptyset}$ do not make any predictions.

Property 2. Given a diagnosis problem $\langle \mathcal{O}, B, T^{\models}, T^{\not\models} \rangle$, a set of diagnoses $\mathbf{D}$, and a query X:

If the oracle gives the answer *1* then the set of rejected diagnoses is $\mathbf{D}^{\neg\mathbf{X}}$ and the set of remaining diagnoses is $\mathbf{D}^{\mathbf{X}} \cup \mathbf{D}^{\emptyset}$.

If the oracle gives the answer *0* then the set of rejected diagnoses is $\mathbf{D}^{\mathbf{X}}$ and the set of remaining diagnoses is $\mathbf{D}^{\neg\mathbf{X}} \cup \mathbf{D}^{\emptyset}$.

For our first example let us consider three possible queries X_1, X_2 and X_3 (see Table 2). For each query we can partition a set of diagnoses $\mathbf{D}$ into three sets $\mathbf{D}^{\mathbf{X}}$, $\mathbf{D}^{\neg\mathbf{X}}$ and $\mathbf{D}^{\emptyset}$. Using this data and the heuristic given above we can determine that asking the oracle if $\mathcal{O}_t \models C(w)$ is the best query, as two diagnoses from the set $\mathbf{D}$ are removed regardless of the answer.

Let us assume that $\mathcal{D}_1$ is the target diagnosis, then an oracle will answer 0 to our question (i.e. $\mathcal{O}_t \not\models C(w)$). Given this feedback we can decide that $\mathcal{O}_t \models B(w)$ is the next best query, which is also answered with 0 by the oracle. Consequently, we identified that $\mathcal{D}_1$ is the only remaining minimal diagnosis. More generally, if n is the number of diagnoses and we can split the set of diagnoses in half by each query then the minimum number of queries is $log_2 n$. However, if the probabilities of diagnoses are known we can reduce this number of queries by using two effects: (1) We can exploit diagnoses probabilities to asses the probabilities of answers and the change in information content after an answer is given. (2) Even if there are multiple diagnoses in the set of remaining diagnoses we can stop further query generation if one diagnosis is highly probable and all other remaining diagnoses are highly improbable.

Query	$\mathbf{D}^{\mathbf{X}}$	$\mathbf{D}^{\neg\mathbf{X}}$	$\mathbf{D}^{\emptyset}$
$X_1 : \{B(w)\}$	$\{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_1\}$	$\emptyset$
$X_2 : \{C(w)\}$	$\{\mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_1, \mathcal{D}_2\}$	$\emptyset$
$X_3 : \{Q(w)\}$	$\{\mathcal{D}_4\}$	$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$	$\emptyset$

Table 2. Possible queries in Example 1

Example 2 Consider an ontology $\mathcal{O}$ with the terminology $\mathcal{T}$:

$$ax_1 : A_1 \sqsubseteq A_2 \sqcap M_1 \sqcap M_2 \qquad ax_4 : M_2 \sqsubseteq \forall s.A \sqcap C$$
$$ax_2 : A_2 \sqsubseteq \neg\exists s.M_3 \sqcap \exists s.M_2 \qquad ax_5 : M_3 \equiv B \sqcup C$$
$$ax_3 : M_1 \sqsubseteq \neg A \sqcap B$$

and the background theory $\mathcal{A} : \{A_1(w), A_1(u), s(u, w)\}$. The ontology is inconsistent and includes two minimal conflict sets: $\{\langle ax_1, ax_3, ax_4 \rangle, \langle ax_1, ax_2, ax_3, ax_5 \rangle\}$. To restore consistency, the user should modify all axioms of at least one minimal diagnosis:

$$\mathcal{D}_1 : [ax_1] \quad \mathcal{D}_2 : [ax_3] \quad \mathcal{D}_3 : [ax_4, ax_5] \quad \mathcal{D}_4 : [ax_4, ax_2]$$

Following the same approach as in the first example, we compute entailments for each ontology $\mathcal{O}_i = \mathcal{O} \setminus \mathcal{D}_i$ for all minimal diagnoses $\mathcal{D}_i \in \mathbf{D}$. To construct a query we select a $\mathbf{D}^{\mathbf{X}} \subset \mathbf{D}$ and determine the common set X of concept instantiations and concept subsumption axioms, which are entailed by each $\mathcal{O}_i = \mathcal{O} \setminus \mathcal{D}_i$, where $\mathcal{D}_i \in \mathbf{D}^{\mathbf{X}}$. If the set X is empty, the query is rejected. For each accepted query the remaining diagnoses $\mathcal{D}_j \in \mathbf{D} \setminus \mathbf{D}^{\mathbf{X}}$ are partitioned into three sets $\mathbf{D}^{\mathbf{X}}$, $\mathbf{D}^{\neg\mathbf{X}}$, and $\mathbf{D}^{\emptyset}$ as defined above. If the the ontology $\mathcal{O}_j = \mathcal{O} \setminus \mathcal{D}_j$ is inconsistent with X then we add $\mathcal{D}_j$ to the

Query	$\mathbf{D}^{\mathbf{X}}$	$\mathbf{D}^{\neg\mathbf{X}}$	$\mathbf{D}^{\emptyset}$
$X_1 : \{B \sqsubseteq M_3\}$	$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_4\}$	$\{\mathcal{D}_3\}$	$\emptyset$
$X_2 : \{B(w)\}$	$\{\mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_2\}$	$\{\mathcal{D}_1\}$
$X_3 : \{M_1 \sqsubseteq B\}$	$\{\mathcal{D}_1, \mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_2\}$	$\emptyset$
$X_4 : \{M_1(w), M_2(u)\}$	$\{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_1\}$	$\emptyset$
$X_5 : \{A(w)\}$	$\{\mathcal{D}_2\}$	$\{\mathcal{D}_3, \mathcal{D}_4\}$	$\{\mathcal{D}_1\}$
$X_6 : \{M_2 \sqsubseteq D\}$	$\{\mathcal{D}_1, \mathcal{D}_2\}$	$\emptyset$	$\{\mathcal{D}_3, \mathcal{D}_4\}$
$X_7 : \{M_3(u)\}$	$\{\mathcal{D}_4\}$	$\emptyset$	$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$

Table 3. Possible queries in Example 2

set $\mathbf{D}^{\neg\mathbf{X}}$. In the case when $\mathcal{O}_j \cup \{\neg X\}$ is inconsistent $\mathcal{D}_j$ is added to $\mathbf{D}^{\mathbf{X}}$. Otherwise we add $\mathcal{D}_j$ to the set $\mathbf{D}^{\emptyset}$.

For instance, ontologies $\mathcal{O}_i = \mathcal{O} \setminus \mathcal{D}_i$ obtained for diagnoses $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}_4$ have the following set of common entailments:

$$X'_4 : \{A_1 \sqsubseteq A_2, A_1 \sqsubseteq M_1, A_1 \sqsubseteq M_2, A_2(u), M_1(u), M_2(u), A_2(w), M_1(w)\} \quad (1)$$

Since the set X'_4 is not empty it is considered as the query and the set $\mathbf{D}^{\mathbf{X}}$ includes three elements $\{\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}$. The ontology $\mathcal{O} \setminus \mathcal{D}_1 \cup \{X'_4\}$ is inconsistent therefore the set $\mathbf{D}^{\neg\mathbf{X}} = \{\mathcal{D}_1\}$ and the set $\mathbf{D}^{\emptyset} = \emptyset$. However, a query need not include all of these axioms. If a query X' partitions the set of diagnoses into $\mathbf{D}^{\mathbf{X}}$, $\mathbf{D}^{\neg\mathbf{X}}$ and $\mathbf{D}^{\emptyset}$ and there exists an irreducible set $X \subset X'$ which preserves the partition then it is sufficient to query X. In our example, the set X'_4 can be reduced to its subset $X_4 : \{M_1(w), M_2(u)\}$. If there are multiple subsets that preserve the partition we select one with minimal cardinality. For query generation we investigate all possible subsets of $\mathbf{D}$. This is feasible since we consider only the n most probable minimal diagnoses (e.g. $n = 12$) during query generation and selection.

The possible queries presented in Table 3 partition the set of diagnoses $\mathbf{D}$ in a way that makes the application of myopic strategies, such as split-in-half, inefficient. A greedy algorithm based on such a heuristic would select the first query X_1 as the next query, since there is no query that cuts the set of diagnoses in half. If $\mathcal{D}_4$ is the target diagnosis then X_1 will be positively evaluated by an oracle (see Fig. 1). On the next iteration the algorithm would also choose a suboptimal query since there is no partition that divides the diagnoses $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_4$ into two equal groups. Consequently, it selects the first untried query X_2. The oracle answers positively, and the algorithm identifies query X_4 to differentiate between $\mathcal{D}_1$ and $\mathcal{D}_4$.

However, in real-world settings the assumption that all axioms fail with the same probability is rarely the case. For example, Roussey et al. [12] present a list of "anti-

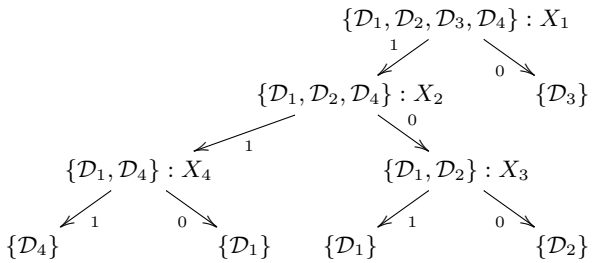

Fig. 1. Greedy algorithm

patterns". Each anti-pattern is a set of axioms, like $\{C1 \sqsubseteq \forall R.C2, C1 \sqsubseteq \forall R.C3, C2 \equiv \neg C3\}$, that correspond to a minimal conflict set. The study performed by the authors shows that such conflict sets occur often in practice and therefore can be used to compute probabilities of diagnoses.

The approach that we follow in this paper was suggested by Rector at al. [10] and considers the syntax of the description logics, such as quantifiers, conjunction, negation, etc., rather than axioms to describe a failure pattern. For instance, if a user modifies a quantifier of one of the roles to restore coherency, then we can assume that axioms including universal quantifier are more probable to fail than the other ones. In [10] the authors report that in most cases inconsistent ontologies were created because users (a) mix up $\forall r.S$ and $\exists r.S$, (b) mix up $\neg\exists r.S$ and $\exists r.\neg S$, (c) mix up $\sqcup$ and $\sqcap$, (d) wrongly assume that classes are disjoint by default or overuse disjointness, (e) wrongly apply negation. Observing that misuses of quantifiers are more likely than other failure patterns one might find that the axioms ax_2 and ax_4 are more likely to be faulty than ax_3 (because of the use of quantifiers), whereas ax_3 is more likely to be faulty than ax_5 and ax_1 (because of the use of negation). Therefore, diagnosis $\mathcal{D}_2$ is the most probable one, followed closely by $\mathcal{D}_4$ although it is a double fault diagnosis. $\mathcal{D}_1$ and $\mathcal{D}_3$ are significantly less probable because ax_1 and ax_5 have a significantly lower fault probability than ax_3. A detailed justification based on probability is given in the next section.

Taking into account the information about user faults provided in [10], it is almost useless to ask query X_1 because it is highly probable that the target diagnosis is either $\mathcal{D}_2$ or $\mathcal{D}_4$ and therefore it is highly probable that the oracle will respond with 1. Instead, asking X_3 is more informative because given any possible answer we can exclude one of the highly probable diagnoses, i.e. either $\mathcal{D}_2$ or $\mathcal{D}_4$. If the oracle responds to X_3 with 0 then $\mathcal{D}_2$ is the only remaining diagnosis. However, if the oracle responds with 1, diagnoses $\mathcal{D}_4$, $\mathcal{D}_3$, and $\mathcal{D}_1$ remain, where $\mathcal{D}_4$ is significantly more probable compared to diagnoses $\mathcal{D}_3$ and $\mathcal{D}_1$. We can stop, since the difference between the probabilities of the diagnoses is high enough such that $\mathcal{D}_1$ can be accepted as the target diagnosis. In other situations additional questions may be required. This strategy can lead to a substantial reduction in the number of queries compared to myopic approaches as we will show in our evaluation.

Note that in real-world application scenarios failure patterns and their probabilities can be discovered by analyzing actions of a user in an ontology editor, like Protégé, while debugging an ontology or just repairing an inconsistency/incoherency. In this case it is possible to "personalize" the debugging algorithm such that it will prefer user-specific faults.

3 Entropy-based query selection

To select the best query we make the assumption that knowledge is available about the a-priori failure probabilities in specifying axioms. Such probabilities can be estimated either by studies such as [10, 12] or can be personalized by observing the typical failures of specific users working with an ontology development tool. In the last case an ontology editor should just save logs of debugging sessions, as well as user actions taken to restore the consistency/coherency of an ontology. Such observations can be then used to

identify typical failures of a particular user. Using observations about failure patterns, for instance obtained from an ontology editor as described above, we can calculate the initial probability of each axiom $p(ax_i)$ containing a failure. If no information about failures is available then the debugger can initialize all probabilities $p(ax_i)$ with some small number.

Given the failure probabilities $p(ax_i)$ of axioms, the diagnosis algorithm first calculates the a-priori probability $p(\mathcal{D}_j)$ that $\mathcal{D}_j$ is the target diagnosis. Since all axioms fail independently, this probability can be computed as [8]:

$$p(\mathcal{D}_j) = \prod_{ax_n \in \mathcal{D}_j} p(ax_n) \prod_{ax_m \notin \mathcal{D}_j} 1 - p(ax_m) \tag{2}$$

The prior probabilities for diagnoses are then used to initialize an iterative algorithm that includes two main steps: (a) selection of the best query and (b) update of the diagnoses probabilities given the query feedback.

According to information theory the best query is the one that, given the answer of an oracle, minimizes the expected entropy of a the set of diagnoses [8]. Let $p(X_i = v_{ik})$ where $v_{i0} = 0$ and $v_{i1} = 1$ be the probability that query X_i is answered with either 0 or 1. Let $p(\mathcal{D}_j|X_i = v_{ik})$ be the probability of diagnosis $\mathcal{D}_j$ after the oracle answers $X_i = v_{ik}$. The expected entropy after querying X_i is:

$$H_e(X_i) = \sum_{k=0}^{1} p(X_i = v_{ik}) \times - \sum_{\mathcal{D}_j \in \mathbf{D}} p(\mathcal{D}_j|X_i = v_{ik}) \log_2 p(\mathcal{D}_j|X_i = v_{ik})$$

The query which minimizes the expected entropy is the best one based on a one-step-look-ahead information theoretic measure. This formula can be simplified to the following score function [8] which we use to evaluate all available queries and select the one with the minimum score to maximize information gain:

$$sc(X_i) = \sum_{k=0}^{1} p(X_i = v_{ik}) \log_2 p(X_i = v_{ik}) + p(\mathbf{D_i^\emptyset}) + 1 \tag{3}$$

where $\mathbf{D_i^\emptyset}$ is the set of diagnoses which do not make any predictions for the query X_i. $p(\mathbf{D_i^\emptyset})$ is the total probability of the diagnoses that predict no value for the query X_i. Since, for a query X_i the set of diagnoses $\mathbf{D}$ can be partitioned into the sets $\mathbf{D^{X_i}}$, $\mathbf{D^{\neg X_i}}$ and $\mathbf{D_i^\emptyset}$, the probability that an oracle will answer a query X_i with either 1 or 0 can be computed as:

$$p(X_i = v_{ik}) = p(\mathbf{S_{ik}}) + p(\mathbf{D_i^\emptyset})/2 \tag{4}$$

where $\mathbf{S_{ik}}$ corresponds to the set of diagnoses that predicts the outcome of a query, e.g. $\mathbf{S_{i0}} = \mathbf{D^{\neg X_i}}$ for $X_i = 0$ and $\mathbf{S_{i1}} = \mathbf{D^{X_i}}$ in the other case. Under the assumption that *both outcomes are equally likely* the probability that a set of diagnoses $\mathbf{D_i^\emptyset}$ predicts $X_i = v_{ik}$ is $p(\mathbf{D_i^\emptyset})/2$.

Since by Definition 1 each diagnosis is a unique partition of all axioms in an ontology $\mathcal{O}$ into correct and faulty, we consider all diagnoses as mutually exclusive events. Therefore the probabilities of their sets can be calculated as:

$$p(\mathbf{D_i^\emptyset}) = \sum_{\mathcal{D}_j \in \mathbf{D_i^\emptyset}} p(\mathcal{D}_j) \qquad p(\mathbf{S_{ik}}) = \sum_{\mathcal{D}_j \in \mathbf{S_{ik}}} p(\mathcal{D}_j)$$

Given the feedback v of an oracle to the selected query X_s, i.e. $X_s = v$ we have to update the probabilities of the diagnoses to take the new information into account. The update is made using Bayes' rule for each $\mathcal{D}_j \in \mathbf{D}$:

$$p(\mathcal{D}_j|X_s = v) = \frac{p(X_s = v|\mathcal{D}_j)p(\mathcal{D}_j)}{p(X_s = v)} \tag{5}$$

where the denominator $p(X_s = v)$ is known from the query selection step (Equation 4) and $p(\mathcal{D}_j)$ is either a prior probability (Equation 2) or is a probability calculated using Equation 5 during the previous iteration of the debugging algorithm. We assign $p(X_s = v|\mathcal{D}_j)$ as follows:

$$p(X_s = v|\mathcal{D}_j) = \begin{cases} 1, & \text{if } \mathcal{D}_j \text{ predicted } X_s = v; \\ 0, & \text{if } \mathcal{D}_j \text{ is rejected by } X_s = v; \\ \frac{1}{2}, & \text{if } \mathcal{D}_j \in \mathbf{D}_\mathbf{s}^\emptyset \end{cases}$$

Example 1 (continued) Suppose that the debugger is not provided with any information about possible failures and therefore it assumes that all axioms fail with the same probability $p(ax_i) = 0.01$. Using Equation 2 we can calculate probabilities for each diagnosis. For instance, $\mathcal{D}_1$ suggests that only one axiom ax_1 should be modified by the user. Hence, we can calculate the probability of diagnosis D_1 as follows $p(\mathcal{D}_1) = p(ax_1)(1 - p(ax_2))(1 - p(ax_3))(1 - p(ax_4)) = 0.0097$. All other minimal diagnoses have the same probability, since every other minimal diagnosis suggests the modification of one axiom. To simplify the discussion we only consider minimal diagnoses for the query selection. Therefore, the prior probabilities of the diagnoses can be normalized to $p(\mathcal{D}_j) = p(\mathcal{D}_j)/\sum_{\mathcal{D}_j \in \mathbf{D}} p(\mathcal{D}_j)$ and are equal to 0.25.

Given the prior probabilities of the diagnoses and a set of queries (see Table 2) we evaluate the score function (Equation 3) for each query. E.g. for the first query $X_1 : \{B(w)\}$ the probability $p(\mathbf{D}^\emptyset) = 0$ and the probabilities of both the positive and negative outcomes are: $p(X_1 = 1) = p(\mathcal{D}_2) + p(\mathcal{D}_3) + p(\mathcal{D}_4) = 0.75$ and $p(X_1 = 0) = p(\mathcal{D}_1) = 0.25$. Therefore the query score is $sc(X_1) = 0.1887$.

The scores computed during the initial stage (see Table 4) suggest that X_2 is the best query. Taking into account that $\mathcal{D}_1$ is the target diagnosis the oracle answers 0 to the query. The additional information obtained from the answer is then used to update the probabilities of diagnoses using the Equation 5. Since $\mathcal{D}_1$ and $\mathcal{D}_2$ predicted this answer, their probabilities are updated, $p(\mathcal{D}_1) = p(\mathcal{D}_2) = 1/p(X_2 = 1) = 0.5$. The probabilities of diagnoses $\mathcal{D}_3$ and $\mathcal{D}_4$ which are rejected by the outcome are also updated, $p(\mathcal{D}_3) = p(\mathcal{D}_4) = 0$.

On the next iteration the algorithm recomputes the scores using the updated probabilities. The results show that X_1 is the best query. The other two queries X_2 and X_3 are

Query	Initial score	$X_2 = 1$
$X_1 : \{B(w)\}$	0.1887	**0**
$X_2 : \{C(w)\}$	**0**	1
$X_3 : \{Q(w)\}$	0.1887	1

Table 4. Expected scores for queries ($p(ax_i) = 0.01$)

Query	Initial score
$X_1 : \{B(w)\}$	**0.250**
$X_2 : \{C(w)\}$	0.408
$X_3 : \{Q(w)\}$	0.629

Table 5. Expected scores for queries ($p(ax_1) = 0.025$, $p(ax_2) = p(ax_3) = p(ax_4) = 0.01$)

Answers	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
Prior	0.0970	0.5874	0.0026	0.3130
$X_3 = 1$	0.2352	0	0.0063	0.7585
$X_3 = 1, X_4 = 1$	0	0	0.0082	0.9918
$X_3 = 1, X_4 = 1, X_1 = 1$	0	0	0	1

Table 6. Probabilities of diagnoses after answers

Queries	Initial	$X_3 = 1$	$X_3 = 1, X_4 = 1$
$X_1 : \{B \sqsubseteq M_3\}$	0.974	0.945	**0.931**
$X_2 : \{B(w)\}$	0.151	0.713	1
$X_3 : \{M_1 \sqsubseteq B\}$	**0.022**	1	1
$X_4 : \{M_1(w), M_2(u)\}$	0.540	**0.213**	1
$X_5 : \{A(w)\}$	0.151	0.713	1
$X_6 : \{M_2 \sqsubseteq D\}$	0.686	0.805	1
$X_7 : \{M_3(u)\}$	0.759	0.710	0.970

Table 7. Expected scores for queries

irrelevant since no information will be gained if they are performed. Given the negative feedback of an oracle to X_1, we update the probabilities $p(\mathcal{D}_1) = 1$ and $p(\mathcal{D}_2) = 0$. In this case the target diagnosis $\mathcal{D}_1$ was identified using the same number of steps as the split-in-half heuristic.

However, if the first axiom is more likely to fail, e.g. $p(ax_1) = 0.025$, then the first query will be $X_1 : \{B(w)\}$ (see Table 5). The recalculation of the probabilities given the negative outcome $X_1 = 0$ sets $p(\mathcal{D}_1) = 1$ and $p(\mathcal{D}_2) = p(\mathcal{D}_3) = p(\mathcal{D}_4) = 0$. Therefore the debugger identifies the target diagnosis only in one step.

Example 2 (continued) Suppose that in ax_4 the user specified $\forall s.A$ instead of $\exists s.A$ and $\neg\exists s.M_3$ instead of $\exists s.\neg M_3$ in ax_2. Therefore $\mathcal{D}_4$ is the target diagnosis. Moreover, the debugger is provided with observations of three types of failures: (1) conjunction/disjunction occurs with probability $p_1 = 0.001$, (2) negation $p_2 = 0.01$, and (3) restrictions $p_3 = 0.05$. Using the probability addition rule for non-mutually exclusive events we can calculate the probability of the axioms containing an error: $p(ax_1) = 0.0019$, $p(ax_2) = 0.1074$, $p(ax_3) = 0.012$, $p(ax_4) = 0.051$, and $p(ax_5) = 0.001$. These probabilities are exploited to calculate the prior probabilities of the diagnoses (see Table 6) and to initialize the query selection process.

On the first iteration the algorithm determines that X_3 is the best query and asks an oracle whether $\mathcal{O}_t \models M_1 \sqsubseteq B$ is true or not (see Table 7). The obtained information is then used to recalculate the probabilities of the diagnoses and to compute the next best query X_4, and so on. The query process stops after the third query, since $\mathcal{D}_4$ is the only diagnosis that has the probability $p(\mathcal{D}_4) > 0$.

Given the feedback of the oracle $X_4 = 1$ for the second query, the updated probabilities of the diagnoses show that the target diagnosis has a probability of $p(\mathcal{D}_4) = 0.9918$ whereas $p(\mathcal{D}_3)$ is only 0.0082. In order to reduce the number of queries a user can specify a threshold, e.g. $\sigma = 0.95$. If the probability of some diagnosis is greater than this threshold, the query process stops and returns the most probable diagnosis. Note, that even after the first answer $X_3 = 1$ the most probable diagnosis $\mathcal{D}_3$ is three times more

Algorithm 1: Ontology debugging algorithm

Input: ontology $\mathcal{O}$, set of background axioms B, set of fault probabilities for axioms FP, maximum number of most probable minimal diagnoses n, acceptance threshold σ
Output: a diagnosis $\mathcal{D}$

$DP \leftarrow \emptyset$; $DS \leftarrow \emptyset$; $T^{\models} \leftarrow \emptyset$; $T^{\not\models} \leftarrow \emptyset$; $\mathbf{D} \leftarrow \emptyset$; $s \leftarrow 0$;
while `belowThreshold`$(DP, \sigma) \wedge s \neq 1$ **do**
  $\mathbf{D} \leftarrow$ `getDiagnoses`(`HS-Tree`$(\mathcal{O}, B \cup T^{\models}, T^{\not\models}, n)$);
  $DS \leftarrow$ `computeDataSet`$(DS, \mathbf{D})$;
  $DP \leftarrow$ `computePriors`$(\mathbf{D}, FP)$;
  $DP \leftarrow$ `uptateProbablities`$(DP, DS, T^{\models}, T^{\not\models})$;
  $s \leftarrow$ `getMinimalScore`(DS, DP);
  $\left\langle X, \mathbf{D}^{\mathbf{X}}, \mathbf{D}^{\neg\mathbf{X}} \right\rangle \leftarrow$ `selectQuery`(DS, s);
  if `getAnswer`$(\mathcal{O}_t \models X)$ **then** $T^{\models} \leftarrow T^{\models} \cup X$;
  else $T^{\not\models} \leftarrow T^{\not\models} \cup \neg X$;
return `mostProbableDiagnosis`$(\mathbf{D}, DP)$;

likely than the second most probable diagnosis D_1. Given such a great difference we could suggest to stop the query process after the first answer. Thus, in this example the debugger requires less queries than the split-in-half heuristic.

4 Implementation details

The ontology debugger (Algorithm 1) takes an ontology $\mathcal{O}$ as input. Optionally, a user can provide a set of axioms B that are known to be correct, a set FP of fault probabilities for axioms $ax_i \in \mathcal{O}$, a maximum number n of most probable minimal diagnoses that should be considered by the algorithm, and a diagnosis acceptance threshold σ. The fault probabilities of axioms are computed as described by exploiting knowledge about typical user errors. Parameters n and σ are used to speed up the computations. In Algorithm 1 we approximate the set of the n most probable diagnoses with the set of the n most probable *minimal* diagnoses, i.e. we neglect non-minimal diagnoses which are more probable than some minimal ones. This approximation is correct, under a reasonable assumption that probability of each axiom $p(ax_i) < 0.5$. In this case for every non-minimal diagnosis ND, a minimal diagnosis $\mathcal{D} \subset ND$ exists which from Equation 2 is more probable than ND. Consequently the query selection algorithm operates on the set of minimal diagnoses instead of all diagnoses (including non-minimal ones). However, the algorithm can be adapted with moderate effort to also consider non-minimal diagnoses.

We implemented the computation of diagnoses following the approach proposed by Friedrich et al. [3]. The authors employ the combination of two algorithms, QUICKXPLAIN [5] and HS-TREE [11]. The latter is a search algorithm that takes an ontology $\mathcal{O}$, a set of correct axioms, a set of axioms $T^{\not\models}$ which must not be implied by the target ontology, and the maximal number of most probable minimal diagnoses n as an input. HS-TREE implements a breadth-first search strategy to compute a set of minimal hitting sets from the set of all minimal conflicts in $\mathcal{O}$. As suggested in [3] it ignores all branches of the search tree that correspond to hitting sets inconsistent with at least one

element of $T^{\not\models}$. HS-TREE terminates if either it identifies the n most probable minimal diagnoses or there are no further diagnoses which are more probable than the already computed ones. Note, HS-TREE often calculates only a small number of minimal conflict sets in order to generate the n most probable minimal hitting sets (i.e. minimal diagnoses), since only a subset of all minimal diagnoses is required.

The search algorithm computes minimal conflicts using QUICKXPLAIN. This algorithm, given a set of axioms AX and a set of correct axioms B returns a minimal conflict set $CS \subseteq AX$, or $\emptyset$ if axioms $AX \cup B$ are consistent. Minimal conflicts are computed on-demand by HS-TREE while exploring the search space. The set of minimal hitting sets returned by HS-TREE is used by GETDIAGNOSES to create a set $\mathbf{D}$ with at most n minimal diagnoses.

At the beginning of the main loop the algorithm calls COMPUTEDATASET function to generate a set of ontologies $\mathbf{O} : \{\mathcal{O}_i\}$ for each diagnosis $\mathcal{D}_i \in \mathbf{D}$ by removing all elements of a diagnosis from $\mathcal{O}$. The algorithm uses this set to generate data sets like the ones presented in Tables 2 and 3. For each ontology $\mathcal{O}_i \in \mathbf{O}$ the algorithm gets a set of entailments from the reasoner and associates them with the corresponding diagnosis $\mathcal{D}_i$. The algorithm uses the set of diagnoses/entailments pairs to compute the set of queries. For each query X_i it partitions the set $\mathbf{D}$ into $\mathbf{D}^{\mathbf{X_i}}$, $\mathbf{D}^{\neg\mathbf{X_i}}$ and $\mathbf{D}^{\emptyset}_{\mathbf{i}}$, as defined in Section 2. Then X_i is iteratively reduced by applying QUICKXPLAIN such that sets $\mathbf{D}^{\mathbf{X_i}}$ and $\mathbf{D}^{\neg\mathbf{X_i}}$ are preserved.

In the next step COMPUTEPRIORS computes prior probabilities for a set of diagnoses given the fault probabilities of the axioms contained in FP. To take past answers into account the algorithm updates the prior probabilities of the diagnoses by evaluating Equation 5 for each diagnosis in $\mathbf{D}$ (UPDATEPROBABILITIES). All data required for the update is stored in sets DS, $T^{\models}$, and $T^{\not\models}$.

The function GETMINIMALSCORE evaluates the scoring function (Equation 3) for each element of DS and returns the minimal score.

In the query-selection phase the algorithm selects a set of axioms that should be evaluated by an oracle. SELECTQUERY retrieves a triple $\langle X, \mathbf{D}^{\mathbf{X}}, \mathbf{D}^{\neg\mathbf{X}} \rangle \in DS$ that corresponds to the best (minimal) score s. The set of axioms X is then presented to the oracle. If there are multiple queries with a minimal score SELECTQUERY returns the triple where X has the smallest cardinality in order to reduce the answering effort.

Depending on the answer of the oracle, the algorithm extends either set $T^{\models}$ or $T^{\not\models}$. This is done to exclude corresponding diagnoses from the results of HS-TREE in further iterations. Note, the algorithm can be easily extended to allow the oracle to reject a query if the answer is unknown. In this case the algorithm proceeds with the next best query until no further queries are available.

The algorithm stops if there is a diagnosis probability above the acceptance threshold σ or if no query can be used to differentiate between the remaining diagnoses (i.e. all scores are 1). The most probable diagnosis is then returned to the user. If it is impossible to differentiate between a number of highly probable minimal diagnoses, the algorithm returns a set that includes all of them.

5 Evaluation

The evaluation of our approach was performed using generated examples and real-world ontologies presented in Table 1. We employed generated examples to perform

controlled experiments where the number of minimal diagnoses and their cardinality could be varied to make the identification of the target diagnosis more difficult. The main goal of the experiment using ontologies is to demonstrate applicability of our approach in the real-world settings.

For the first test we created a generator which takes a consistent and coherent ontology, a set of fault patterns together with their probabilities, the minimum number of minimal diagnoses m, and the required minimum cardinality of these minimal diagnoses $|\mathcal{D}_t|$ as inputs. The output was an alteration of the input ontology for which at least the given number of minimal diagnoses with the required cardinality exist. In order to introduce inconsistencies and incoherences, the generator applied fault patterns randomly to the input ontology depending on their probabilities.

In this experiment we took five fault patterns from a case study reported by Rector at al. [10] and assigned fault probabilities according to their observations of typical user errors. Thus we assumed that in cases (a) and (b) (see Section 2, when an axiom includes some roles (i.e. property assertions), axiom descriptions are faulty with a probability of 0.025, in cases (c) and (d) 0.01 and in case (e) 0.001. In each iteration the generator randomly selected an axiom to be altered and applied a fault pattern to this axiom. Next it selected another axiom using the concept taxonomy and altered it correspondingly to introduce an incoherency/inconsistency. The fault patterns were randomly selected in each step using the probabilities given above.

For instance, given the description of a randomly selected concept A and the fault pattern "misuse of negation", we added the construct $\sqcap\neg X$ to the description of A, where X is a new concept name. Next, we randomly selected concepts B and S such that $S \sqsubseteq A$ and $S \sqsubseteq B$ and added $\sqcap X$ to the description of B. During the generation process, we applied the HS-TREE algorithm after each introduction of a incoherency/inconsistency to control two parameters: the minimum number of minimal diagnoses in the ontology and their minimum cardinality. The generator continued to introduce incoherences/inconsistencies until the specified parameter values were reached. For instance, if the minimum number of minimal diagnoses equals to $m = 6$ and their cardinality to $|\mathcal{D}_t| = 4$, then the generated ontology will include at least 6 diagnoses of cardinality 4 and some additional number of diagnoses of higher cardinalities.

The resulting faulty ontology as well as the fault patterns and their probabilities were inputs for the ontology debugger. The acceptance threshold σ was set to 0.95 and the number of most probable minimal diagnoses n was set to 12. One of the minimal diagnoses with the required cardinality was randomly selected as the target diagnosis. Note, the target ontology is not equal to the original ontology, but rather is a corrected version of the altered one, in which the faulty axioms were repaired by replacing them with their original (correct) versions according to the target diagnosis. The tests were done on ontologies bike2 to bike9, bcs3, galen and galen2 from Racer's benchmark suite[1].

The average results of the evaluation performed on each test suite (depicted in Fig. 2) show that the entropy-based approach outperforms the split-in-half method described in Section 2 as well as random query selection by more than 50% for the $|\mathcal{D}_t| = 2$ case due to its ability to estimate the probabilities of diagnoses. On average the algorithm required 8 seconds to generate a query. Figure 2 also shows that the car-

[1] http://www.racer-systems.com/products/download/benchmark.phtml

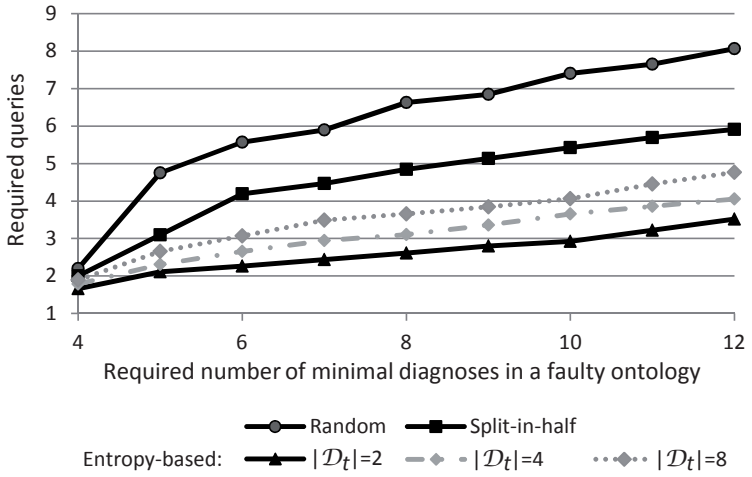

Fig. 2. Number of queries required to select the target diagnosis $\mathcal{D}_t$ with threshold $\sigma = 0.95$. Random and "split-in-half" are shown for the cardinality of minimal diagnoses $|\mathcal{D}_t| = 2$.

dinality of the target diagnosis increases as the number of required queries increases. This holds for the random and split-in-half methods (not depicted) as well. However, the entropy-based approach is still better than the split-in-half method even for diagnoses with increasing cardinality. The approach required more queries to discriminate between high cardinality diagnoses because the prior probabilities of these diagnoses tend to converge.

In the tests performed on the real-world ontologies we initialized the input parameters n and σ of Algorithm 1 with the same values as in the test with generated examples. Also we used the same five fault patterns together with their probabilities as given above. Before the experiment each ontology was analyzed by the HS-TREE algorithm and all minimal diagnoses of these ontologies were identified. In each test for a given ontology we selected randomly one of its minimal diagnoses as the target one and applied our approach using both split-in-half and entropy-based strategies. The evaluation of queries was done automatically by verifying if a query is also entailed by the target ontology obtained by removing all axioms of the target diagnosis from the input ontology. For each ontology we performed 20 tests and on each iteration the target diagnosis was randomly reselected.

The results of this experiment are presented in Tables 8 and 9 and show that in terms of queries, the entropy-based approach outperformed split-in-half. As the number of diagnoses grew we observed that the difference between the two strategies increased. In the best case for the entropy-based strategy, when the target diagnoses were assigned a high a-priori fault probability, the number of queries was usually twice as low as required by the split-in-half strategy. Also in the worst case, when the target diagnoses were assigned a low a-priori fault probability, the entropy-based strategy performed better than split-in-half, because it was able to adapt the a-posteriori fault probabilities using Bayes rule and the oracle's feedback to queries. In this case the entropy-based strategy corresponds to active learning [14] applied to learn fault probabilities which

	Ontology	Split-in-half min	Split-in-half max	Split-in-half avg	Entropy-based min	Entropy-based max	Entropy-based avg
1.	Chemical	3	4	3	1	3	2
2.	Sweet-JPL	4	5	4	1	4	2
3.	University	7	9	8	2	7	4
4.	Tambis	8	10	8	2	7	5
5.	Economy	10	12	11	3	10	6
6.	Transport	11	14	12	4	11	7

Table 8. Number of queries required to identify a target diagnosis

	Ontology	Diagnoses 12	Diagnoses all	Query avg
1.	Chemical	0,97	1,39	1,50
2.	Sweet-JPL	31,97	36,47	5,48
3.	University	0,27	0,61	1,12
4.	Tambis	80,29	286,11	3,91
5.	Economy	8,33	55,70	1,87
6.	Transport	6,70	99,02	2,39

Table 9. Time in seconds required to calculate 12 first and all minimal diagnoses as well as an average time used to generate a query

is not exploited in the split-in-half strategy. The more queries are asked, the better the entropy-based method can predict the target diagnosis.

6 Related work

To the best of our knowledge no sequential ontology debugging methods (neither employing split-in-half nor entropy-based methods) have been proposed to debug faulty ontologies so far. Diagnosis methods for ontologies are introduced in [13, 6, 3]. Ranking of diagnoses and proposing a target diagnosis is presented in [7]. This method uses a number of measures such as: (a) the frequency with which an axiom appears in conflict sets, (b) impact on an ontology in terms of its "lost" entailments when some axiom is modified or removed, (c) ranking of test cases, (d) provenance information about the axiom, and (e) syntactic relevance. All these measures are evaluated for each axiom in a conflict set. The scores are then combined in a rank value which is associated with the corresponding axiom. These ranks are then used by a modified HS-TREE algorithm that identifies diagnoses with a minimal rank. In this work no query generation and selection strategy is proposed if the target diagnosis cannot be determined reliably with the given a-priori knowledge. In our work additional information is acquired until the target diagnosis can be identified with confidence. In general, the work of [7] can be combined with the one presented in this paper as axiom ranks can be taken into account together with other observations while calculating the prior probabilities of the diagnoses.

The idea of selecting the next best query based on the expected entropy was exploited in the generation of decisions trees [9] and further refined for selecting measurements in the model-based diagnosis of circuits [8]. We extended these methods to query selection in the domain of ontology debugging.

7 Conclusions

In this paper we presented an approach to the sequential diagnosis of ontologies. We showed that the axioms generated by classification and realization can be used to build queries which differentiate between diagnoses. To rank the utility of these queries we employ knowledge about typical user errors in ontology axioms. Based on the likelihood of an ontology axi om containing an error we predict the information gain produced by a query result, enabling us to select the next best query according to a one-step-lookahead entropy-based scoring function. We outlined the implementation of a sequential debugging algorithm and compared our proposed method with a split-in-half strategy. Our experiments showed a significant reduction in the number of queries required to identify the target diagnosis.

References

1. Ceraso, J., Provitera, A.: Sources of error in syllogistic reasoning. Cognitive Psychology 2(4), 400–410 (1971)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press, New York, 2nd edn. (2007)
3. Friedrich, G., Shchekotykhin, K.: A General Diagnosis Method for Ontologies. In: 4th International Semantic Web Conference (ISWC-05). pp. 232–246. Springer (2005)
4. Haarslev, V., Müller, R.: RACER System Description. In: 1st International Joint Conference on Automated Reasoning (IJCAR-01). pp. 701–705. Springer (2001)
5. Junker, U.: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In: Association for the Advancement of Artificial Intelligence (AAAI-04). pp. 167–172. AAAI (2004)
6. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all Justifications of OWL DL Entailments. In: 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC-07) pp. 267–280. Springer (2007)
7. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B.: Repairing Unsatisfiable Concepts in OWL Ontologies. In: 3rd European Semantic Web Conference (ESWC-06). pp. 170–184. Springer (2006)
8. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artificial Intelligence 32(1), 97–130 (Apr 1987)
9. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (Mar 1986)
10. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW-04). pp. 63–81. Springer (2004)
11. Reiter, R.: A Theory of Diagnosis from First Principles. Artificial Intelligence 23, 57–95 (Apr 1987)
12. Roussey, C., Corcho, O., Vilches-Blázquez, L.M.: A catalogue of OWL ontology antipatterns. In: 5th International Conference On Knowledge Capture (K-CAP-09). pp. 205–206. ACM (2009)
13. Schlobach, S., Huang, Z., Cornet, R., Harmelen, F.: Debugging Incoherent Terminologies. Journal of Automated Reasoning 39(3), 317–349 (Oct 2007)
14. Settles, B.: Active Learning Literature Survey. Computer sciences technical report 1648, University of Wisconsin-Madison (2009)
15. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (Jun 2007)