A New Expert Questioning Approach to More Efficient Fault Localization in Ontologies

Patrick Rodler[0000−0001−8178−3333] and Michael Eichholzer

Alpen-Adria Universität Klagenfurt, 9020 Klagenfurt, Austria
patrick.rodler@aau.at
michael.eichholzer@aon.at

Abstract. When ontologies reach a certain size and complexity, faults such as inconsistencies, unsatisfiable classes or wrong entailments are hardly avoidable. Locating the faulty axioms that cause these faults is a hard and time-consuming task. Addressing this issue, several techniques for semi-automatic fault localization in ontologies have been proposed. Often, these approaches involve a human expert who provides answers to system-generated questions about the intended (correct) ontology in order to reduce the possible fault locations. To suggest as informative questions as possible, existing methods draw on various algorithmic optimizations as well as heuristics. However, these computations are often based on certain assumptions about the interacting user.
In this work, we characterize and discuss different user types and show that existing approaches do not achieve optimal efficiency for all of them. As a remedy, we suggest a new type of expert question which aims at fitting the answering behavior of all analyzed experts. Moreover, we present an algorithm to optimize this new query type which is fully compatible with the (tried and tested) heuristics used in the field. Experiments on faulty real-world ontologies show the potential of the new querying method for minimizing the expert consultation time, independent of the expert type. Besides, the gained insights can inform the design of interactive debugging tools towards better meeting their users' needs.

Keywords: Ontology Debugging · Interactive Debugging · Fault Localization · Sequential Diagnosis · Expert Questions · Ontology Quality Assurance · Ontology Repair · Test-Driven Debugging

1 Introduction

As Semantic Web technologies have become widely adopted in, e.g., government, security and health applications, the quality assurance of the data, information and knowledge used by these applications is a critical requirement. At the core of semantic web technologies, ontologies are a means to represent knowledge in a formal, structured and human-readable way, with a well-defined semantics. As ontologies are often developed and cured in a collaborative way by numerous contributors [41, 39] possibly not sharing their conceptualization of the domain of interest, are merged by automated alignment tools [13], reach sizes and complexities exceeding human reasoning and understanding capabilities [5], or use expressive logical formalisms such as OWL 2 [6], faults occur regularly during the evolution of ontologies [13, 32, 3, 17]. Since one of the

major benefits of ontologies is the capability of using them to perform logical reasoning and thereby solve relevant problems, faults that affect the ontology's semantics are of particular concern for semantic applications. Specifically, such faults may cause the ontology, e.g., to become inconsistent, include unsatisfiable classes or feature wrong entailments.

One important step towards the repair of such faults is the *localization* of the responsible faulty axioms. To handle nowadays ontologies with often thousands of axioms, several fault localization approaches [13, 33, 10, 14] have been proposed to semi-automatically assist humans in this complex and time-consuming task. These approaches, which are mainly based on the *model-based diagnosis* framework [18, 12], use the faulty ontology along with additional specifications to reason about different fault assumptions. Such fault assumptions are called *diagnoses* if they are consistent with all given specifications. The specifications usually comprehend some requirements to the correct ontology, e.g., in the form of *logical properties* (e.g., consistency, coherency), and/or in terms of necessary and forbidden entailments. The latter are usually referred to as *positive and negative test cases* [4, 33, 31].

Research on model-based diagnosis has brought up various algorithms [18, 12, 10, 19, 13, 34] for computing and ranking diagnoses; however, a frequent problem is that a high number of competing diagnoses might exist where all of them lead to repaired ontologies with necessarily different semantics [19]. Finding the correct diagnosis (pinpointing the actually faulty axioms) is thus crucial for successful and sustainable repair. Since it is a mentally-demanding task for humans to recognize and reason about entailments and non-entailments [7] of the ontology under particular fault assumptions, interactive techniques[1] [33, 19] have been developed to undertake this task and relieve the user as much as possible. What remains to be accomplished by the interacting human—usually an ontology engineer or a domain expert (referred to as *expert* in the sequel)—is the answering of a series of *queries* about the intended ontology that are shown to them by the system. Roughly, that means the user has to classify certain axioms as either entailments (positive test cases) or non-entailments (negative test cases) of the intended ontology. A concrete implementation of such a query-based fault localization approach is *OntoDebug*[2] [30], a plug-in for the popular ontology editor *Protégé* [15].

Several evaluations [33, 34, 26] have shown the feasibility and usefulness of query-based fault localization, and its efficiency has been improved by various algorithmic optimizations [9, 35, 22, 27] and the use of heuristics [33, 28, 21, 25, 20] for the selection of the most informative questions to ask an expert. However, the used heuristics, algorithms and optimization criteria are based on certain assumptions about the question answering behavior of experts. In this work, we critically discuss existing approaches with regard to these assumptions. Particularly, we characterize different types of experts and show that not all of them are equally well accommodated by current querying approaches. That is, we observe that the necessary expert interaction cost to locate the ontology's faults is significantly influenced by the way queries posed by the debugging system are answered. To overcome this issue, we propose a new way of user interaction

[1] Depending on the community, these techniques are referred to as Sequential Diagnosis and Interactive (or: Test-Driven) Ontology (or: Knowledge Base) Debugging.

[2] All information about OntoDebug can be found at http://isbi.aau.at/ontodebug/

that serves all discussed expert types equally well and moreover increases the expected amount of information relevant for fault localization obtained from the expert per asked axiom. In addition, we present a polynomial time and space algorithm to generate and optimize the newly suggested type of question in terms of the well-understood and proven heuristics used in the field.

The main idea behind the new approach is to restrict questions—which are, for quite natural reasons, *sets of* axioms in existing methods—only to *single* axioms, as usually done in *sequential diagnosis* applications [12, 37], where systems different from ontologies (e.g., digital circuits) are analyzed and such singleton queries are the natural choice. That is, experts are asked single axioms at a time instead of getting batch queries which (possibly) include multiple axioms. Experiments on real-world faulty ontologies manifest the reasonability and usefulness of the new approach. Specifically, we find that, in more than two thirds of the studied cases, the new querying technique is superior to existing ones in terms of minimizing the number of required expert inputs, regardless of the type of expert. In addition, the time for the determination of the best next query is reduced by at least 80 % in all investigated cases when using singleton queries instead of existing techniques.

The rest of the work is organized as follows. In Section 2, we give a short introduction to query-based fault localization in ontologies, before we challenge certain assumptions made by state-of-the-art approaches in the field in Section 3. We describe our proposed approach in Section 4, where we also discuss its pros and cons, and elaborate an algorithm for the computation of the suggested new query type. Our experiments and the obtained results are explicated in Section 5. Finally, we point to open questions and both interesting and promising future research issues in Section 6, before we summarize the conclusions from this work in Section 7.

2 Query-Based Fault Localization in Ontologies

We briefly recap basic technical concepts used in works on ontology fault localization, based on [19, 33]. As a running example we reuse the example presented in [25].

Fault Localization Problem Instance. We assume a faulty ontology to be given by the finite set of axioms $\mathcal{O} \cup \mathcal{B}$, where $\mathcal{O}$ includes the *possibly faulty* axioms and $\mathcal{B}$ the *correct* (background knowledge) axioms, and $\mathcal{O} \cap \mathcal{B} = \emptyset$ holds. This partitioning of the ontology means that faulty axioms must be sought only in $\mathcal{O}$, whereas $\mathcal{B}$ provides the fault localization context. At this, $\mathcal{B}$ can be useful to achieve a fault search space restriction (if parts of the faulty ontology are marked correct) or a higher fault detection rate (if external approved knowledge is taken into account, which may point at otherwise undetected faults). Besides logical properties such as consistency and coherency, requirements to the intended (correct) ontology can be formulated as a set of test cases [4], analogously as it is common practice in software engineering [2]. In particular, we distinguish between two types of test cases, positive (set P) and negative (set N) ones. Each test case is a set (interpreted as conjunction) of axioms; positive ones $p \in P$ *must* be and negative ones $n \in N$ *must not* be entailed by the intended ontology. We call $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ an *(ontology) fault localization problem instance (FPI)*.

Example 1. Consider the following ontology with the terminology $\mathcal{T}$:

$$\{ \quad ax_1 : ActiveResearcher \sqsubseteq \exists writes.(Paper \sqcup Review)\ , \\ ax_2 : \exists writes.\top \sqsubseteq Author\ , \quad ax_3 : Author \sqsubseteq Employee \sqcap Person \quad \}$$

and assertions $\mathcal{A} : \{ax_4 : ActiveResearcher(ann)\}$. To locate faults in the terminology while accepting as correct the assertion and stipulating that Ann is not necessarily an employee (negative test case $n_1 : \{Employee(ann)\}$), one can specify the following FPI: $fpi_{ex} := \langle \mathcal{T}, \mathcal{A}, \emptyset, \{n_1\} \rangle$. □

Fault Hypotheses. Let $U_P := \bigcup_{p \in P} p$ and $\mathbf{C}_\perp := \{C \sqsubseteq \perp \mid C$ named class in $\mathcal{O}, \mathcal{B}$ or $P\}$. Given that the ontology, along with the positive test cases, is inconsistent or incoherent, i.e. $\mathcal{O} \cup \mathcal{B} \cup U_P \models x$ for some $x \in \{\perp\} \cup \mathbf{C}_\perp$, or some negative test case is entailed, i.e. $\mathcal{O} \cup \mathcal{B} \cup U_P \models n$ for some $n \in N$, some axioms in $\mathcal{O}$ must be accordingly modified or deleted to enable the formulation of the intended ontology. We call such a set of axioms $\mathcal{D} \subseteq \mathcal{O}$ a *diagnosis* for the FPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ iff $(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup U_P \not\models x$ for all $x \in N \cup \{\perp\} \cup \mathbf{C}_\perp$. $\mathcal{D}$ is a *minimal diagnosis* iff there is no diagnosis $\mathcal{D}' \subset \mathcal{D}$. We call $\mathcal{D}^*$ *the actual diagnosis* iff all $ax \in \mathcal{D}^*$ are faulty and all $ax \in \mathcal{O} \setminus \mathcal{D}^*$ are correct. For efficiency and to suggest changes to the faulty ontology that preserve as much of its meaning as possible, fault localization approaches usually restrict their focus to the computation of minimal diagnoses.

Example 2. For $fpi_{ex} = \langle \mathcal{O}, \mathcal{B}, P, N \rangle$ from Example 1, $\mathcal{O} \cup \mathcal{B} \cup U_P$ entails the negative test case $n_1 \in N$, i.e. that Ann is an employee. The reason is that according to $ax_1 (\in \mathcal{O})$ and $ax_4 (\in \mathcal{B})$, Ann writes some paper or review since she is an active researcher. Due to the additional $ax_2 (\in \mathcal{O})$, Ann is also an author because she writes something. Finally, since Ann is an author, she must be both an employee and a person, as postulated by $ax_3 (\in \mathcal{O})$. Hence, $\mathcal{D}_1 : [ax_1]$, $\mathcal{D}_2 : [ax_2]$, $\mathcal{D}_3 : [ax_3]$ are (all the) minimal diagnoses for fpi_{ex}, as the deletion of any $ax_i \in \mathcal{O}$ breaks the unwanted entailment n_1. □

Eliminating Wrong Fault Hypotheses. The main idea model-based diagnosis systems use for fault localization, i.e., to find the actual diagnosis among the set of all (minimal) diagnoses, is that different fault assumptions have (necessarily [19]) different semantic properties in terms of entailments and non-entailments. This fact can be exploited to distinguish between diagnoses by asking an expert whether a (set of) axiom(s) Q, which is entailed by some and inconsistent with some other fault assumptions, must be correct or not. More formally, given a known set of minimal diagnoses $\mathbf{D}$, a *(normal) query* (wrt. $\mathbf{D}$) is a set of axioms Q that rules out at least one diagnosis in $\mathbf{D}$, both if Q is classified as a positive test case and if Q is classified as a negative test case. That is, at least one $\mathcal{D}_i \in \mathbf{D}$ is not a diagnosis for $\langle \mathcal{O}, \mathcal{B}, P \cup \{Q\}, N \rangle$ and at least one diagnosis $\mathcal{D}_j \in \mathbf{D}$ is not a diagnosis for $\langle \mathcal{O}, \mathcal{B}, P, N \cup \{Q\} \rangle$. A query Q corresponds to the question "Is (the conjunction of axioms in) Q an entailment of the intended ontology?". The expert who provides answers to queries can be modeled as a function $\mathsf{expert} : \mathbf{Q} \to \{y, n\}$ where $\mathbf{Q}$ is the query space; $\mathsf{expert}(Q) = y$ iff the answer to the question is positive, else $\mathsf{expert}(Q) = n$.

Every set of axioms X partitions any set of diagnoses $\mathbf{D}$ for an FPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ into three subsets—the diagnoses predicting that X is a positive test case (set $\mathbf{D}_X^+ \subseteq \mathbf{D}$), the ones predicting that X is a negative test case (set $\mathbf{D}_X^- \subseteq \mathbf{D}$), and the ones that do not predict any classification for X (set $\mathbf{D}_X^0 \subseteq \mathbf{D}$). More specifically, among the diagnoses in $\mathbf{D}$, $\mathbf{D}_X^+$ comprises exactly the diagnoses that are no diagnoses for

$\langle \mathcal{O}, \mathcal{B}, P, N \cup \{Q\} \rangle$, $\mathbf{D}^-_X$ those that are no diagnoses for $\langle \mathcal{O}, \mathcal{B}, P \cup \{Q\}, N \rangle$, and $\mathbf{D}^0_X$ all remaining ones. A partition $\mathfrak{P}$ of $\mathbf{D}$ into three sets is called *q-partition* iff there is a query Q wrt. $\mathbf{D}$ such that $\mathfrak{P} = \left\langle \mathbf{D}^+_Q, \mathbf{D}^-_Q, \mathbf{D}^0_Q \right\rangle$. According to the definition of a query, it holds that Q is a query iff both $\mathbf{D}^+_Q$ and $\mathbf{D}^-_Q$ are non-empty sets. The notion of a q-partition is leveraged by current approaches for *query generation* [26], *query verification* [33] and *query quality estimation* [27, 21].

Example 3. Let the known set of diagnoses for fpi_{ex} be $\mathbf{D} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ (see Example 2). One query wrt. $\mathbf{D}$ is, e.g., $Q_1 := \{ActiveResearcher \sqsubseteq Author\}$. Because, *(i)* adding Q_1 to P yields that the removal of $\mathcal{D}_1$ or $\mathcal{D}_2$ from $\mathcal{O}$ no longer breaks the unwanted entailment $Employee(ann)$, i.e., $\mathcal{D}_1, \mathcal{D}_2$ are no longer minimal diagnoses, *(ii)* moving Q_1 to N means that $\mathcal{D}_3$ is not a minimal diagnosis anymore, as, to prevent the entailment of (the new negative test case) Q_1, at least one of ax_1, ax_2 must be deleted. The resulting q-partition for Q_1 is thus $\langle \mathbf{D}^+_{Q_1}, \mathbf{D}^-_{Q_1}, \mathbf{D}^0_{Q_1} \rangle = \langle \{\mathcal{D}_3\}, \{\mathcal{D}_1, \mathcal{D}_2\}, \emptyset \rangle$. Note, e.g., $Q_2 := \{Author \sqsubseteq Person\}$, is not a query since no diagnosis in $\mathbf{D}$ is invalidated upon assigning Q_2 to P, i.e., a positive answer does not give any useful information for diagnoses discrimination. Intuitively, this is because Q_2 does not contribute to the violation of n_1 (in fact, the other "part" $Author \sqsubseteq Employee$ of ax_3 does so). □

Problem Definition. The query-based ontology fault localization problem (QFL) is to find for an FPI a series of questions to an expert, the answers of which lead to a single possible remaining fault assumption. The optimization version of the problem includes the additional goal to minimize the effort of the expert. Formally:

Problem 1 ((Optimal) QFL). **Given:** FPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$. **Find:** (Minimal-cost) series of queries $Q_1, \ldots, Q_k$ s.t. there is only one minimal diagnosis for $\langle \mathcal{O}, \mathcal{B}, P \cup P', N \cup N' \rangle$ where P' (N') is the set of all positively (negatively) answered queries, i.e., $P' := \{Q_i \mid 1 \leq i \leq k, \mathsf{expert}(Q_i) = y\}$ and $N' := \{Q_i \mid 1 \leq i \leq k, \mathsf{expert}(Q_i) = n\}$.

Note, there is no unified definition of the cost of a solution to the QFL problem. Basically, any function mapping $Q_1, \ldots, Q_k$ to a non-negative real number is possible. We pick up on this discussion again in Sec. 3.

Example 4. Let the actual diagnosis be $\mathcal{D}_3$, i.e. ax_3 is the (only) faulty axiom in $\mathcal{O}$ (intuition: an author is not necessarily employed, but might be, e.g, a freelancer). Then, given fpi_{ex} as an input, solutions to Problem 1, yielding the final diagnosis $\mathcal{D}_3$, are, e.g., $P' = \emptyset, N' = \{\{\exists writes.\top \sqsubseteq Employee\}, \{Author \sqsubseteq Employee\}\}$ or $P' = \{\{ActiveResearcher \sqsubseteq Author\}\}, N' = \emptyset$. Measuring the querying cost by the number of queries, the latter solution (cost: 1) is optimal, the former (cost: 2) not. □

3 Discussion of Query-based Fault Localization Approaches

In this section we analyze existing approaches regarding the assumptions they make about (the query answering behavior of) the interacting user, their properties resulting from natural design choices, as well as optimization criteria they consider.

Assumptions about Query Answering. All proposed approaches drawing on the interactive methodology described in Sec. 2 make the assumption *during their computations and optimizations* that the expert evaluates each query as a whole. That is, they perform an assessment of the query effect or (information) gain *based on two possible outcomes* (y and n). However, in fact, since queries might contain multiple axioms, the feedback of an expert to a query might take a multitude of different shapes. Because, the expert might not view the query as an atomic question, but at the axiom level, i.e., inspecting axioms one-by-one. Clearly, to answer the query $Q = \{ax_1, \ldots, ax_m\}$ positively—i.e., that the conjunction of the axioms $ax_1, \ldots, ax_m$ is an entailment of the intended ontology—one needs to scrutinize and approve the entailment of all single axioms. To negate the query Q, in contrast, it suffices to detect one of the m axioms in Q which is not an entailment of the intended ontology. In this latter case, however, we might reasonably assume the interacting expert to be able to name (at least this) one *specific* axiom $ax^* \in Q$ that is not an intended entailment. We might think of ax^* as a "witness of the falsehood of the query". This additional information—beyond the mere negative answer n indicating that some *undefined* query axiom must not be entailed—justifies the addition of $n^* := \{ax^*\}$, instead of Q, to the negative test cases. Please note that n^* provides stronger information than Q, and thus potentially rules out more diagnoses. The reason is that each diagnosis that entails Q (i.e., is invalidated given the negative test case Q) particularly entails ax^* (i.e., is definitely invalidated given the negative test case n^*). Apart from the scenario where experts provide just a falsehood-witness in the negative case, they might give even more information. For instance, an expert could walk through the query axioms until either a non-entailed one is found or all axioms have been verified as intended entailments. In this case, there might as well be some entailed axioms encountered before the first non-entailed one is detected. The set of these entailed axioms could then be added to the positive test cases—in addition to the negative test case n^*. Alternatively, the expert might also continue evaluating axioms after recognizing the first non-entailed axiom ax^*, in this vein providing the classification of all single query axioms in Q.

Based on this discussion, we might—besides the *query-based* expert that answers queries as a whole, exactly as specified by the `expert` function defined in Sec. 2—characterize (at least) three different types of *axiom-based* experts which supply information beyond the mere n label of the query in the negative case:[3]

- *Minimalist:* Provides exactly one $ax^* \in Q$ which is not entailed by the intended ontology.
- *Pragmatist:* Provides the first found axiom $ax^* \in Q$ that is not entailed by the intended ontology, and all axioms evaluated as entailments of the intended ontology until ax^* was found.
- *Maximalist:* Provides the classification of each axiom in Q as either an entailment or a non-entailment of the intended ontology.

Consequently: *(i)* In general, without knowing the answering type of the interacting expert in advance, the binary query evaluation conducted in existing works is only an

[3] Note that a positive query answer (y) implicitly provides *axiom-level* information, i.e., the positive classification of all query-axioms. Therefore, the discussed expert types differ only in their query negation behavior.

approximation. *(ii)* Also if the expert type is known, it is an open issue which form of interaction can exploit the expert knowledge most beneficially and economically. Our experimental evaluations reported in Sec. 5 shall confirm (i) and bring light to (ii).

Natural Design Choices. As explicated in Sec. 2, the principle behind queries is the comparison of entailments and non-entailments resulting from different fault assumptions (diagnoses). In existing works [33, 28], this is often done by computing common entailments (of specific types)—e.g., subsumption and assertion axioms resulting from classification and realization reasoning services [1]—for some diagnoses and verify whether some other diagnosis becomes inconsistent when assuming correct these axioms. At this, it stands to reason to use and further process *all* entailments returned by the reasoner. Moreover, the fewer entailments are used, the higher is the chance that these are entailed by all (known) diagnoses and hence do not constitute a query. Besides, assuming a *query-based* expert (see above), query selection heuristics [33, 28, 21, 25] can be optimized to a higher degree due to the simple fact that a larger allowed cardinality of queries implies a larger search space for queries. For these reasons, it is quite natural to specify queries as *sets of* axioms.

Optimization Criteria. The meaning of "minimal-cost" in Problem 1 might be defined in different ways. Most existing works on query-based fault localization, e.g., [33, 30, 28, 19]— especially in the empirical analyses they present—specify the cost of a solution $Q_1, \ldots, Q_k$ to the QFL problem to be *the number of* queries, i.e., k. The underlying assumption in this case is that each two queries mean the same (answering) cost for an expert. Given that queries might include fewer or more axioms of lower or higher (syntactic or semantic) complexity, we argue that this cost measure might be too coarse-grained to capture the effort for an interacting expert in a realistic way. Instead, it might be better suited to measure the costs at the axiom level. However, a fundamental problem with a minimization of the axiom level costs is the need to compute the specific query axioms for multiple (or all) queries, which generally involves high computation costs in terms of a high number of reasoner calls. A remedy to this problem and a two-staged technique to minimize both the number of queries and the costs at the axiom level is suggested by [26]. However, the user type taken as a basis for these optimizations is again the query-based one (see above).

4 New Approach to Expert Interaction

4.1 Idea

In the light of the issues pointed out in Sec. 3 and following quite straightforward from the given argumentation, we propose a new way of expert interaction for fault localization in ontologies, namely to abandon "batch-queries" including multiple axioms and to focus on so-called *singleton queries* instead. That is, we suggest to restrict queries to only single-axiom questions. Formally:

Definition 1 (Singleton Query). *Let* $\mathbf{D}$ *be a set of diagnoses for an FPI* $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$. *Then,* Q *is a singleton query (wrt.* $\mathbf{D}$*) iff* Q *is a query (wrt.* $\mathbf{D}$*) and* $|Q| = 1$.

4.2 Properties

The *advantages* of singleton queries are the following:

- *Maximally-fine granularity of optimization loop:* Each atomic expert input (i.e., each classified axiom) can be directly taken into account to optimize further computations and expert interactions. Simply put, each axiom the expert is asked to classify is a function of *all* so-far classified axioms.
- *Smaller search space:* There are fewer singleton queries than there are general queries. Therefore, the worst-case search costs are lower for singleton queries.
- *Realistic query assessment:* For singleton queries, the binary-outcome assessment performed by the discussed approaches is exact, plausible and not just an approximation of the possible real cases—independent of the expert (type). The reason is that there *are* exactly two possible outcomes, namely y (query axiom added to P) and n (query axiom added to N).
- *Direct re-use of existing works:* Concepts (e.g., heuristics) and techniques (e.g., search algorithms) defined for queries can be immediately re-used for singleton queries, because each singleton query *is* a (specific) query.
- *Unique optimization criterion:* Query-number minimization and (axiom-based) answering-cost minimization coincide for singleton queries. This unifies the two competing and arguable views on the query optimization problem.
- *More informative expert feedback:* Negative answers to singleton queries provide more information than negative answers to normal queries as the former imply that we know one axiom which is wrong *for sure*, whereas the latter just tell us that *one of a set of* axioms is not true. Therefore, singleton queries, by their nature, implicitly appoint how they are answered, independent of the expert (type). Because all discussed expert types coincide for singleton queries.

On the downside, the smaller search space—apart from the better worst-case query optimization complexity—can be seen as a *disadvantage* as well. Because soundness of the query search is more difficult to obtain, i.e., more considerations and computations than for normal queries are required to ensure that the search outcome is indeed a *singleton* query. For instance, after having optimized a predefined heuristic measure for some query candidate (set of axioms) to a sufficient degree, existing approaches [33, 20] post-process this candidate by a query-size minimization step. This step, however, does not guarantee the reduction to a single axiom. Thus, beside all the mentioned advantages of singleton queries, an algorithmic and computational challenge towards their efficient generation and optimization remains to be solved.

4.3 Generation and Optimization

As a first step in this direction we suggest an algorithm that, given a set of diagnoses $\mathbf{D}$, finds the (next) heuristically-optimal[4] singleton query $Q \subseteq \mathcal{O}$ (wrt. $\mathbf{D}$) to ask the

[4] The *global* optimization of query costs is proven NP-hard [8] (even without considering the reasoning complexity for diagnosis and query generation). Hence, the best that methods can achieve is to optimize some heuristic in each query computation iteration. To this end, a one-

expert. In this vein, the algorithm can be used in each iteration of a sequential fault localization session. Such a session is characterized by a loop involving a re-iteration of the three phases *(1)* fault hypotheses generation (computation of diagnoses), *(2)* query generation and optimization, and *(3)* query answering and incorporation of the newly acquired test case(s), until only one diagnosis is left.[5] By the theory of model-based diagnosis [18, 12], this final diagnosis necessarily includes the faulty axioms explaining all observed problems (e.g., inconsistency, unsatisfiable classes, wrong entailments) of the ontology. Thus, used for query computation in a sequential session, our algorithm presented below will deliver a (heuristics-based approximation of the optimal) series of ontology axioms such that the assignment of each of these axioms to either the positive or the negative test cases solves Problem 1.

The works of [20, 27] serve as a theoretical and algorithmic basis for our method. In fact, we slightly extend the theory and adapt the algorithm presented there to accommodate singleton queries. First, we briefly review the existing query computation and optimization algorithm for normal queries, and next we present our adaptations to it.

Query Computation and Optimization for Normal Queries (Recap). Basically, the algorithm [26] is subdivided into two stages, namely a search for a heuristically-optimal q-partition $\mathfrak{P}$ (stage 1) and a search for a cost-optimal query (set of axioms) for this fixed q-partition $\mathfrak{P}$ (stage 2). At this, the first stage serves the purpose of optimizing a heuristic function, e.g., the expected information gain [12, 33], that aims at minimizing the expected *number of queries*. The goal of the second stage is to minimize the *cost for query answering* based on some axiom-based cost measure, e.g., the number of axioms.

Stage 1: Here, a heuristic search is performed. Such a search is characterized [29] by a start state, a goal state, a successor function (what are the immediate neighbor states of a given state?) as well as a heuristic function (what is the expected utility of visiting a given state?). Originally, the "depth- first, local best-first backtracking" algorithm works as follows. *(Depth-first):* Starting from the initial partition $\langle \emptyset, \mathbf{D}, \emptyset \rangle$ (start state), the search proceeds downwards by "shifting" diagnoses from the middle ($\mathbf{D}^-$) to the left ($\mathbf{D}^+$) part of the q-partition[6] until *(a)* a q-partition with sufficiently optimal heuristic value has been found (goal state), or *(b)* there are no successors of the currently analyzed q-partition. *(Local best-first):* At each current q-partition, the focus moves on to the best *direct* successor q-partition, according to the given heuristic function.[7] *(Backtracking):* The search procedure backtracks in case all successors of a

step-lookahead query evaluation [11] (what is the expected situation after the query has been answered?) is state-of-the-art and also used in this present as well as in existing works. Note the similarity to decision tree learning approaches [16].

[5] Note that this condition must be fulfilled after having obtained the answer to a *finite* number of queries as each query, regardless of its answer, rules out at least one diagnosis (cf. Sec. 2), and the number of diagnoses is bounded by the number of subsets of the *finite* ontology $\mathcal{O}$.

[6] Note, q-partitions with non-empty $\mathbf{D}^0$ (i.e., right) part tend to be unfavorable (see argumentation in [26]) and are thus totally neglected in the q-partition search discussed here for efficiency reasons. So, in the sequel, we will always assume $\mathbf{D}^0 = \emptyset$ for all mentioned q-partitions.

[7] The predicate "local" refers to the fact that the best q-partition to visit next is determined *solely* based on the direct successors of the q-partition.

q-partition have been explored and no goal q-partition has been found yet. In this case, the next-best unexplored sibling of the q-partition will be analyzed next.

The detailed definition of the used successor function is beyond the scope of this work. Therefore, we exemplify the underlying principle through an example [26]:[8]

Example 5. Let a set of minimal diagnoses for an FPI be $\mathbf{D} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\} = \{\{2,3\}, \{2,5\}, \{2,6\}, \{2,7\}, \{1,4,7\}, \{3,4,7\}\}$, where axioms are represented as numbers for simplicity of notation. Be the current q-partition analyzed in the search $\mathfrak{P} = \langle\{\mathcal{D}_5\}, \mathbf{D} \setminus \{\mathcal{D}_5\}, \emptyset\rangle$. Given a q-partition as an input, the goal of the successor function is to output the set of all q-partitions obtainable by *minimal* changes from the input q-partition. These direct successor q-partitions can be computed by means of the notion of a trait. The *traits* for a q-partition $\langle\mathbf{D}^+, \mathbf{D}^-, \emptyset\rangle$ are given by $\mathcal{D}'_i := \mathcal{D}_i \setminus U(\mathbf{D}^+)$ for all $\mathcal{D}_i \in \mathbf{D}^-$. For $\mathfrak{P}$, the traits $\mathcal{D}'_1, \mathcal{D}'_2, \mathcal{D}'_3, \mathcal{D}'_4, \mathcal{D}'_6$ are given by $\{2,3\}$, $\{2,5\}$, $\{2,6\}$, $\{2\}$, $\{3\}$, where, e.g., $\mathcal{D}'_6 = \mathcal{D}_6 \setminus U(\{\mathcal{D}_5\}) = \{3,4,7\} \setminus \{1,4,7\} = \{3\}$. *Successors of a q-partition exist iff there are at least two different subset-minimal traits for this q-partition.* For $\mathfrak{P}$, this holds true, since $\mathcal{D}'_4$ as well as $\mathcal{D}'_6$ are subset-minimal; note, however, that all other traits are not subset-minimal as they are each proper supersets of $\mathcal{D}'_4$ or $\mathcal{D}'_6$. *If successors exist for a q-partition $\mathfrak{P}_r = \langle\mathbf{D}_r^+, \mathbf{D}_r^-, \emptyset\rangle$, then its direct successors are given by the q-partitions resulting from $\mathfrak{P}_r$ by transferring all diagnoses from $\mathbf{D}_r^-$ to $\mathbf{D}_r^+$ which have the same trait and whose trait is subset-minimal among all traits for $\mathfrak{P}_r$.* For $\mathfrak{P}$, this means that there are two direct successors, namely $\langle\{\mathcal{D}_5, \mathcal{D}_4\}, \mathbf{D} \setminus \{\mathcal{D}_5, \mathcal{D}_4\}, \emptyset\rangle$ and $\langle\{\mathcal{D}_5, \mathcal{D}_6\}, \mathbf{D} \setminus \{\mathcal{D}_5, \mathcal{D}_6\}, \emptyset\rangle$. □

Stage 2: In this phase, a query (set of axioms) is sought for the fixed (and already optimal) q-partition returned by stage 1. [26] shows that the queries (comprising ontology axioms) for a q-partition are exactly the hitting sets[9] of all traits for this q-partition. Axiom costs can be minimized by computing hitting sets in best-first order, e.g., by means of the hitting set algorithm presented in [19]. For instance, in order to minimize the number of axioms in the query, a minimum-cardinality-first hitting set computation will do.

Example 6. For the q-partition $\mathfrak{P}$ from Example 5, all subsets of $\{2,3,5,6\}$ that include 2 or 3 are queries. The queries with a minimal number of axioms are $\{2\}$ and $\{3\}$. □

Extension to Singleton Queries. We now present the amendments to the reviewed query computation and optimization algorithm (stages 1 and 2) that are necessary to deal with singleton queries.

To restrict the q-partition search in stage 1 to only q-partitions for singleton queries, we first need a criterion that tells us for which q-partitions associated singleton queries do and do not exist. The following theorem provides such a criterion. The idea is that a singleton query (consisting of an ontology axiom) exists for a q-partition iff all traits for this q-partition include this axiom.

[8] In the sequel, we will use the following abbreviations: Given a collection of sets C, we denote by $U(C)$ the union and by $I(C)$ the intersection of all sets in C.

[9] A set H is a *hitting set* of a collection of sets $C = \{S_1, \ldots, S_n\}$ iff $H \subseteq S_1 \cup \cdots \cup S_n$ and $S_i \cap H \neq \emptyset$ for all $S_i \in C$.

Theorem 1 (Singleton Query Criterion). *Let* $\mathbf{D}$ *be a set of minimal diagnoses for the FPI* $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$ *and* $ax \in \mathcal{O}$*. Then,* $\{ax\}$ *is a singleton query (wrt.* $\mathbf{D}$*) iff there is a q-partition* $\mathfrak{P} = \langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ *(wrt.* $\mathbf{D}$*) such that* $I(\mathbf{D}^-) \setminus U(\mathbf{D}^+) \supseteq \{ax\}$.

Note that Theorem 1, in particular, means that each axiom occurring in some, but not all, (known) diagnoses in $\mathbf{D}$ is a singleton query. However, we want to systematically enumerate an as small as possible number of such queries in a (heuristically) optimal order. Therefore, we next "translate" the above criterion to a successor function that, for any given q-partition, generates all and only singleton query successor q-partitions. Such a function, plugged into the search (stage 1) described above instead of the successor function for normal queries—while re-using everything else of the existing algorithm—yields a sound and complete method for singleton query q-partitions.

Example 7. Recall the diagnoses set $\mathbf{D}$ from Example 5. For this, e.g., $\{7\}$ is a singleton query as there is the q-partition $\mathfrak{P} := \langle \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}, \{\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6\}, \emptyset \rangle$ for which the criterion $I(\mathbf{D}^-) \setminus U(\mathbf{D}^+) = \{7\} \setminus \{2, 3, 5, 6\} \supseteq \{7\}$ holds. However, assuming $\mathbf{D}$ consisted only of, e.g., $\mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6$, $\{7\}$ would not be a (singleton) query (wrt. $\mathbf{D}$). The reason is that a negative answer to it would not invalidate any (known) diagnosis. □

The following matrix-representation for a q-partition's traits is a useful tool towards defining the successor function for singleton query q-partitions.

Definition 2 (Axioms-Traits Matrix (ATM)). *Let* $\mathfrak{P} = \langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ *be a q-partition where* $\mathbf{D}^- = \{\mathcal{D}_{k_1}, \ldots, \mathcal{D}_{k_n}\}$ *and* $\{ax_1, \ldots, ax_m\}$ *be the set of all axioms occurring in the traits* $\mathcal{D}'_{k_1}, \ldots, \mathcal{D}'_{k_n}$ *for* $\mathfrak{P}$*. Then, we call the* $m \times n$*-matrix* $A_{\mathfrak{P}} = (a_{ij})$*, where* $a_{ij} = 1$ *iff* $ax_i \in \mathcal{D}'_{k_j}$ *and* $a_{ij} = 0$ *else, the* axioms-traits matrix (ATM) *for* $\mathfrak{P}$.

Example 8. For the q-partition mentioned in Example 7, the ATM is given by the following matrix. In fact, the matrix represents the statements that axiom $1 \in \mathcal{D}'_5$ (first row), axiom 4 is an element of $\mathcal{D}'_5$, $\mathcal{D}'_6$ (second row), and so on. □

$$\begin{array}{c} \begin{matrix} \mathcal{D}_4 & \mathcal{D}_5 & \mathcal{D}_6 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{matrix} 1 \\ 4 \\ 7 \end{matrix} \end{array}$$

Definition 3 (Domination). *Let* $A_{\mathfrak{P}}$ *be the* $m \times n$ *ATM for a q-partition* $\mathfrak{P}$ *and* $a_{i.}$ *as well as* $a_{j.}$ *be matrix rows where* $1 \leq i, j \leq m$*. Then,* $a_{i.}$ dominates $a_{j.}$ *iff* $a_{ir} = 1$ *for all indices* $r \in \{1, \ldots, n\}$ *for which* $a_{jr} = 1$*. Further,* $a_{i.}$ strictly dominates $a_{j.}$ *iff* $a_{i.}$ *dominates* $a_{j.}$*, but* $a_{j.}$ *does not dominate* $a_{i.}$*. We call a row* superior row *iff it includes at least one 0-entry and is not strictly dominated by any other row.*

Example 9. In the ATM given in Example 8, the second row is the only superior row. The first row is not superior because it includes only 1-entries, and the last row is not since it is dominated by the second one. □

The next theorem states the successor function for singleton queries. Informally, it says that each superior row of a q-partition's ATM represents a singleton query successor q-partition of this q-partition. Each diagnosis associated with a 1-entry in a superior row is an element of the $\mathbf{D}^-$ part of the successor q-partition and all remaining diagnoses in $\mathbf{D}$ are in the $\mathbf{D}^+$ part.

Algorithm 1 (Singleton) Query Selection

Input: set of minimal diagnoses $\mathbf{D}$ for some FPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$, heuristic h_1 (to minimize # of queries) to be optimized in stage 1, heuristic h_2 (to minimize effort per query) to be optimized in stage 2, boolean s affecting the generation of a singleton ($s = true$) or a normal ($s = false$) query
Output: best (singleton) query wrt. h_2 among all queries for the q-partition of $\mathbf{D}$ with best h_1
1: $\mathfrak{P} \leftarrow$ FINDBESTQPARTITION$(\mathbf{D}, h_1, s)$ ▷ stage 1
2: $Q \leftarrow$ FINDBESTQUERYFORQPARTITION$(\mathfrak{P}, h_2, s)$ ▷ stage 2
3: **return** Q

Theorem 2 (Singleton Query Successor Function). *Let $\mathbf{D}$ be a set of minimal diagnoses for an FPI and $\mathfrak{P} = \langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ be a q-partition (wrt. $\mathbf{D}$). Let further $A_{\mathfrak{P}}$ be the ATM associated with $\mathfrak{P}$, and R be the set of the row indices of all superior rows in $A_{\mathfrak{P}}$. Then, the direct singleton query successors of $\mathfrak{P}$ are given by $\left\{ \left\langle \mathbf{D}_i^+, \mathbf{D}_i^-, \emptyset \right\rangle \mid a_{i.} \in R \right\}$ where $\mathbf{D}_i^- = \left\{ \mathcal{D}_{k_j} \mid a_{ij} = 1 \right\}$ and $\mathbf{D}_i^+ = \mathbf{D} \setminus \mathbf{D}_i^-$.*

Example 10. Let us reconsider the q-partition $\mathfrak{P}$ of Example 7. Using Theorem 2 and our observations of Examples 8 and 9, we find that $\langle \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4\}, \{\mathcal{D}_5, \mathcal{D}_6\}, \emptyset \rangle$ is the only singleton query successor q-partition of $\mathfrak{P}$. □

For stage 2 we get—immediately from Theorem 1[10]—that each axiom appearing in all traits of the singleton query q-partition selected in stage 1 is a singleton query:

Corollary 1 (Singleton Query Extraction). *Let $\mathfrak{P} = \langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ be a q-partition that satisfies the criteria given by Theorem 1 and let $A_{\mathfrak{P}}$ be the ATM associated with $\mathfrak{P}$. Then, all singleton queries (consisting of axioms in $\mathcal{O}$) for $\mathfrak{P}$...*

1. *...are given by $\{\{ax\} \mid ax \in I(\mathbf{D}^-) \setminus U(\mathbf{D}^+)\}$.*
2. *...are given exactly by the axioms representing rows with only 1-entries in $A_{\mathfrak{P}}$.*

Example 11. The only singleton query $\{ax\}$ for $ax \in \mathcal{O}$ for the q-partition $\mathfrak{P}$ of Example 7 is $\{7\}$. This can be seen from $\mathfrak{P}$'s ATM shown in Example 8 where the row of 7 is the only row without any 0-entry. □

4.4 Complexity Analysis

The complexity of the suggested algorithm for the generation of a heuristically-optimal singleton query for a given sample of diagnoses is as follows:

Theorem 3 (Complexity). *Let $\mathbf{D}$ be the set of known diagnoses and $n_{\max}$ be the number of axioms in the diagnosis of maximal size in $\mathbf{D}$. Then, Algorithm 1 with setting $s = true$ requires $O(n_{\max}^4 |\mathbf{D}|^3)$ time and $O(n_{\max} |\mathbf{D}|^3)$ space.*

Proof. We first consider the time and then the space complexity.
Time complexity (stage 1): At each node in the search tree a q-partition and a respective ATM must be computed. The construction of a q-partition requires $O(|\mathbf{D}|)$ steps. The creation of an ATM needs one iteration through all (axioms of the) diagnoses in $\mathbf{D}^-$, i.e. $O(n_{\max} |\mathbf{D}|)$ steps.

[10] Note, $I(\mathbf{D}^-) \setminus U(\mathbf{D}^+)$ is exactly the intersection of all traits of the q-partition $\left\langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \right\rangle$.

Successor extraction for the q-partition at each node requires the finding of all superior rows in the ATM. This can be accomplished by checking, for each row, whether it has a 0-entry and whether it is not dominated by any other row. There are $O(n_{\max}|\mathbf{D}|)$ rows (if all diagnoses are disjoint and have equal size $n_{\max}$) and each row has $n_{\max}$ entries. Checking the presence of a 0-entry requires $O(n_{\max})$ checks. Domination can be checked by comparing all (same-indexed) entries of two rows, i.e., by means of $O(n_{\max})$ comparisons. There are $O((n_{\max}|\mathbf{D}|)^2)$ pairs of rows for the domination test. Hence, we need $O(n_{\max}^2|\mathbf{D}| + n_{\max}(n_{\max}|\mathbf{D}|)^2) = O(n_{\max}^3|\mathbf{D}|^2)$ steps for successor computation. Altogether, the time complexity at each node is thus in $O(n_{\max}|\mathbf{D}| + n_{\max}^3|\mathbf{D}|^2) = O(n_{\max}^3|\mathbf{D}|^2)$.

As a consequence of Theorem 1, the number of explored q-partitions in stage 1 is bounded by $|U(\mathbf{D})| \leq \sum_{\mathcal{D}\in\mathbf{D}} |\mathcal{D}| \leq n_{\max}|\mathbf{D}|$, i.e., the q-partition search tree has $O(n_{\max}|\mathbf{D}|)$ nodes.

Consequently, the time complexity of stage 1 is in $O(n_{\max}^4|\mathbf{D}|^3)$.

Time complexity (stage 2): For *one* q-partition $\mathfrak{P}$ (the one selected in stage 1), one (all) singleton queries for $\mathfrak{P}$ can be extracted by scanning all rows of $\mathfrak{P}$'s ATM until one (all) row(s) with only 1-entries are found (Corollary 1). This can be done in $O(n_{\max}|\mathbf{D}|)$ steps (one check for each entry of the ATM). Since all singleton queries can be extracted within this time bound, the best query as per some heuristic can in particular.

Time complexity (overall): So, the time complexity of Algorithm 1 (stage 1 and 2 together) is in $O(n_{\max}^4|\mathbf{D}|^3 + n_{\max}|\mathbf{D}|) = O(n_{\max}^4|\mathbf{D}|^3)$.

Space complexity (stage 1): For each node of the q-partition search tree, we need to store the respective q-partition. The ATM associated with this q-partition needed for successor computation can be computed only at node expansion and does not need to be permanently stored. Also, it can be discarded as soon as all successors have been generated. Note, since the (heuristically-)*best* successor is always chosen as a next node for expansion by the algorithm, such an on-demand computation of the successor q-partitions is not possible. Each q-partition can be stored in $O(n_{\max}|\mathbf{D}|)$ space (which is the space to store all diagnoses in $\mathbf{D}$). Any ATM requires $O(n_{\max}|\mathbf{D}|^2)$ entries because it has at most $n_{\max}|\mathbf{D}|$ rows (if all diagnoses are disjoint and have equal size $n_{\max}$) and at most $|\mathbf{D}|$ columns (there can be no more diagnoses in $\mathbf{D}^-$ than there are in $\mathbf{D}$).

Concerning the number of nodes that must be simultaneously stored during the q-partition search, observe that each successor q-partition results from a q-partition by shifting some diagnosis from its $\mathbf{D}^-$ to its $\mathbf{D}^+$ set. Hence, at most $|\mathbf{D}|$ successors might exist for any q-partition, i.e., the branching factor of the search tree is bounded by $|\mathbf{D}|$. Moreover, the depth of the search tree is bounded by $|\mathbf{D}|$ as well, since along any branch downwards in the search tree diagnoses can only be shifted from $\mathbf{D}^-$ to $\mathbf{D}^+$ (and not vice versa). Since a depth-first search is executed, the space complexity is the product of the branching factor and the maximal tree depth, and is thus given by $O(|\mathbf{D}|^2)$ search tree nodes.

Altogether, the space complexity of stage 1 amounts to the space for a q-partition times the number of q-partitions simultaneously in memory, plus the space for a single ATM (of the currently expanded node). Therefore, stage 1 requires $O(n_{\max}|\mathbf{D}|^3 + n_{\max}|\mathbf{D}|^2) = O(n_{\max}|\mathbf{D}|^3)$ space.

Table 1: Dataset used in the experiments.

j	ontology $\mathcal{O}_j$	$\lvert\mathcal{O}_j\rvert$	expressivity [1)]	#D/min/max [2)]
1	Koala (K) [3)]	42	$\mathcal{ALCON}^{(D)}$	10/1/3
2	University (U) [4)]	50	$\mathcal{SOIN}^{(D)}$	90/3/4
3	MiniTambis (M) [4)]	173	$\mathcal{ALCN}$	48/3/3
4	CMT-Conftool (CC) [5)]	458	$\mathcal{SIN}^{(D)}$	934/2/16
5	Conftool-EKAW (CE) [5)]	491	$\mathcal{ALCH}^{(D)}$	953/3/10
6	Transportation (T) [4)]	1300	$\mathcal{ALCH}^{(D)}$	1782/6/9
7	Economy (E) [4)]	1781	$\mathcal{ALCH}^{(D)}$	864/4/8
8	DBpedia (D) [6)]	7228	$\mathcal{ALCHF}^{(D)}$	7/1/1
9	Opengalen (O) [7)]	9664	$\mathcal{ALEHIF}^{(D)}$	110/2/6
10	Cton (C) [7)]	33203	$\mathcal{SHF}$	15/1/5

Key:
1): Description Logic expressivity [1]
2): #D, min, max denote the number, the min. and max. size of minimal diagnoses for the input FPI.
3): Ontology included in the Protégé Project for educational purposes.
4): Sufficiently complex FPIs (#D $\geq$ 40) used in [33].
5): Hardest FPIs mentioned in [40].
6): Faulty version of the DB-Pedia ontology, downloaded from .
7): Hardest FPIs tested in [33].

Space complexity (stage 2): No additional amount of storage is required for stage 2 because according to Corollary 1 the singleton query can be extracted directly from the ATM of the q-partition selected in stage 1, which however must already be in memory. *Space complexity (overall):* The overall space complexity is thus in $O(n_{\max} * |\mathbf{D}|^3)$. □

Two remarks: First, the input size I of Algorithm 1 is in $O(n_{\max}|\mathbf{D}|)$. So, in terms of I, the time and space complexity is in $O(I^4)$ and in $O(I^3)$, respectively. Second, the number of diagnoses $|\mathbf{D}|$ cannot grow arbitrarily because it is a predefined fixed number that can be set to any (small) value greater or equal 2 [19].

5 Evaluation

Goal. The aim of the following experiments is the analysis of normal queries under different answering conditions (expert types discussed in Sec. 3) and the comparison between normal queries and the proposed singleton queries. Focus of the investigations is the *required effort for the expert* for fault localization and the *query computation time*. Particular questions of interest are:

- Q1 Since existing methods compute and optimize queries based on the assumption of a *query-based expert* (cf. Sec. 3), which implications does a violation of this assumption have on the efficiency of fault localization?
- Q2 Given (a system that computes) a particular type of query, which answering strategy to recommend the interacting expert to pursue?
- Q3 Given a particular (type of) expert, which type of queries to ask them?
- Q4 What is the expected waiting time for the next query in all scenarios?
- Q5 What is better overall, normal or singleton queries?

Dataset, Experiment Settings and Measurements. The dataset of ontologies used in the experiments is given in Tab. 1. All ontologies are real-world examples and are inconsistent and/or incoherent. Each of the ontologies $\mathcal{O}$ was used to specify an FPI $fpi := \langle \mathcal{O}, \emptyset, \emptyset, \emptyset \rangle$, i.e., the background knowledge $\mathcal{B}$ as well as the positive (P) and negative (N) test cases were (initially) empty. Tab. 1 also shows the *diagnostic structure*

(# of axioms $|\mathcal{O}|$, logical expressivity, # and min./max. size of minimal diagnoses) for the considered FPIs. As heuristics (h_1) for stage 1 we used the query selection measures discussed in [21, 25]. For stage 2 we used the number of axioms in the query as a heuristic (h_2). For each FPI and each heuristic h_1 we ran 20 fault localization sessions (each time using a different random specification of the actual diagnosis to be located). The number of diagnoses computed before each query selection (i.e., given as input to Algorithm 1) was set to (maximally) $|\mathbf{D}| = 10$. Since some heuristics (h_1) depend on the diagnoses probabilities, we sampled and assigned uniform random probabilities to diagnoses for each FPI. For each performed fault localization session we measured

M1 the average computation time to find the best next query (*time per Q*),
M2 the average number of q-partitions generated per computed query (*generated QPs per Q*), and
M3 the number of answered queries (*#Q*) as well as
M4 the number of classified query-axioms (*#Ax*)

required until finding the predefined actual diagnosis with certainty.

Representation of Experiment Results. Each of the Figures 9 – 17 provides a per-ontology overview of the observations regarding M1 – M4 we made throughout the experiments, for the ontologies given in Table 1. Specifically, the bars show M3 and M4 for the different expert types, i.e., the minimalist (min), the pragmatist (prag), the maximalist (max), and the query-based expert (q-based), as discussed in Sec. 3. Moreover, the lines report M1 (red line) and M2 (black line). On the x-axis, we have a block showing the values for normal queries (normal Q, left) and a block depicting the measurements for singleton queries (singleton Q, right). In order to not overload the figures and because the observations regarding other heuristics are mostly consistent with the presented ones, Figures 9 – 17 plot only the results for the most-popular heuristics h_1 in the field [33, 28], i.e., ENT (maximize information gain per query), SPL (maximize worst-case diagnoses elimination rate per query) and RIO (optimize balance between ENT and SPL).

Figures 2 – 8 give violin plots for all[11] heuristics in the field [21, 25] that show the difference in query answering effort (M4) between a usage of the best answering strategy for normal queries and the usage of singleton queries. Each violin plot combines a box-plot with a kernel density estimation. In particular, the median is represented by a white dot. If the latter is above (below) the red zero-line, then this means that singleton queries imply less (more) expert effort in the majority of the observed diagnostic sessions. Simply put, the singleton query approach wins on average iff the white dot is above the red line. The additional heuristics not mentioned above that are shown in Figures 9 – 17 are RND (random query selection), BME (select query that maximizes the number of diagnoses that can be eliminated with a probability larger than 0.5), KL (select query with maximal information-theoretic "disagreement" between query-outcome predictions of the known diagnoses) and EMCb (select query that maximizes expected diagnoses elimination rate). For details on these heuristics see [20, 21, 25].

Discussion of Experiment Results. We address questions Q1 – Q5 in turn.

[11] Note that the MPS heuristic is not (directly) applicable to singleton queries and thus omitted.

Ad Q1: As shown by the vertical bars in Figures 9 – 17, if normal queries are answered by an *axiom-based* strategy (min, prag, or max) that provides labels for (some or all) axioms in the query, then the effort for the expert is significantly lower than in case of a *query-based* strategy where an expert just gives a label for the query as such. This effort reduction holds for all ontologies and in terms of both the number of queries (#Q) and the number of checked axioms (#Ax). In fact, this result is not very surprising. The simple reason for it is that each axiom-based method involves strictly more informative answers than a query-based answering style (cf. Sec. 3). However, not all of the axiom-based approaches are equally good, as analyzed in Q2.

Ad Q2: (Normal queries:) As all of the Figures 9 – 17 unequivocally indicate, the pragmatist approach is the optimal choice for normal queries in terms of #Ax. Also wrt. #Q, the pragmatist is the most reasonable expert type, although there are ontologies for which other approaches are better—but, if so, then just marginally. For instance, for ontology C, the maximalist strategy is the best choice when the number of queries should be minimized. The minimalist answering behavior, in contrast, was never the best strategy to minimize #Q in our experiments. However, as argued in Sec. 3, we believe that #Ax is the more reasonable and realistic effort metric. In this view, the pragmatist answering style, where all query-axioms until the first negative one are classified and all others are left unclassified, is clearly the most efficient one.

So, the pragmatist approach appears to be the best trade-off between effort of query answering and achieved gain in terms of diagnoses discrimination. While this result is not self-evident at all, a likely explanation for it is the following. When compared to the maximalist approach, the gain per axiom among the additional axioms classified after having found the first negative axiom is lower than the gain of the first axioms classified (cf. the "law of diminishing returns"). In comparison with the minimalist strategy, it seems that positively classified query-axioms (before the first negative axiom is found), do bring a significant gain as compared to not classifying them.

This matter of fact is quite well exposed in Figures 9 – 17, which show that the number of queries remains approximately the same for all axiom-based answering methods, whereas the number of inspected axioms is minimal for the pragmatist approach.

(Singleton queries:) All four expert types coincide for singleton queries (cf. Sec. 3).

Ad Q3: (Query-based expert:) If the effort metric #Ax is considered, singleton queries are distinctly the interaction method of choice, as clearly evidenced by Figures 9 – 17. The cost overhead in terms of #Ax when relying on normal instead of singleton queries amounts to up to over 200 % (e.g., CC ontology, RIO heuristic). Hence, although normal queries are optimized based on an analysis focusing on the query-based user, singleton queries are drastically more efficient in this scenario. This has two reasons. First, singleton queries are optimized for the query-based expert as well. Because they are—trivially—optimized for all discussed types of users, as all of them behave alike when asked singleton queries. Second, classifying singleton queries brings more information per inspected query-axiom than classifying normal queries. Especially in case the given answer is negative, a singleton query pinpoints a faulty axiom whereas a normal query (including more than one axiom) just indicates that (any) one of its comprised axioms is faulty.

On the other hand, when measuring the effort by #Q, then there are cases where normal queries, and others where singleton queries are better. In concrete terms, singleton queries, on average, prevail over normal ones in all but two (i.e., K and D) of the investigated ontologies. The cause of this lies in the fact that normal (non-singleton) queries provide more information than singleton ones given a positive query answer (multiple vs. a single axiom added to positive test cases), whereas the reverse is true for a negative answer (one *undefined* axiom of multiple asserted wrong vs. one *particular* axiom declared wrong). Obviously, the positive impact of singleton queries in the negation case compared ot normal queries outweighs the reduced gain in the affirmation case in the majority of examined scenarios.

(Axiom-based expert:) Studying Figures 9 – 17 and comparing the best axiom-based answering strategy for normal queries, namely the pragmatist approach (see Q2 above), with singleton queries, we find that, in most cases, the singleton querying method is superior to normal querying as regards #Ax. To illustrate this observation in more detail, Figure 1 shows the incurred overhead in terms of the average #Ax for all heuristics h_1 when using normal queries as compared to singleton queries. For instance, for the three heuristics analyzed in Figures 9 – 17, we find that singleton queries reduce the expert costs on average for 7 of 9 investigated ontologies when using ENT or SPL, and even in 8 of 9 cases for RIO. Averaged over all ontologies, we notice that the highest expected cost reduction by using singleton queries instead of normal ones is achieved by RIO (see rightmost area in Figure 1).

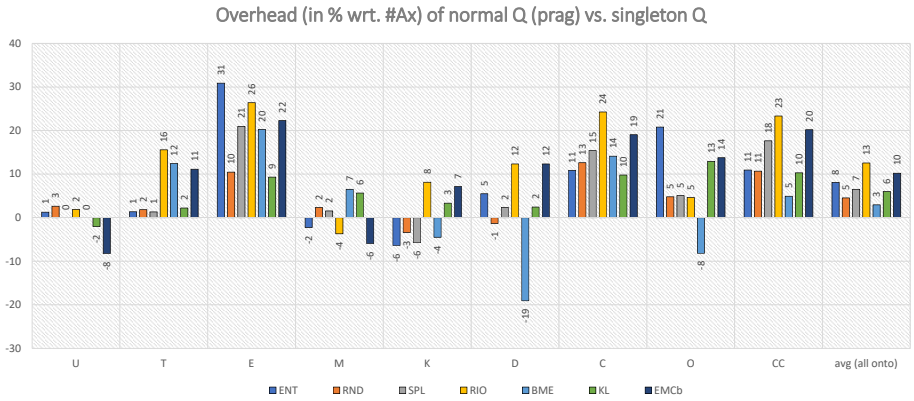

Fig. 1: Bar chart showing the average overhead in % regarding the number of classified axioms (#Ax) per diagnostic session, grouped by ontology (x-axis) and heuristics (different colors).

However, when looking at the single fault localization sessions, there is a significant number of cases where normal queries answered by the pragmatist approach are the best choice wrt. #Ax. This is well illustrated by the violin plots shown in Figures 2 – 8. While singleton queries are the equally good or better choice in the majority of cases (white dot at or above the red line) for *all* ontologies when using the heuristics RND, ENT or KL, for eight of nine ontologies in case of RIO, BME or EMCb, and for seven

of nine in case of SPL, we nevertheless realize that a significant area of almost all violin plots is below the red line. This area denotes the proportion of sessions where normal queries outperformed singleton ones. From this we discern that normal queries *are* a reasonable way of interaction with an expert, but can match up to singleton queries only if the pragmatist answering behavior is given. Hence, existing systems relying on normal queries should advise their users to follow this approach to minimize their debugging time and effort.

On the other hand, when the aim is to minimize #Q, the picture looks at lot different. Here, all axiom-based strategies used in combination with normal queries outperform singleton queries—for all investigated ontologies (see Figures 9 – 17). This situation, however, is absolutely expected and its explanation is straightforward. Because, first, normal queries are computed with the aim to minimize exactly #Q, while being selected from a query space that strictly subsumes the space of singleton queries. Second, normal queries generally comprise multiple axioms, and all axiom-based users classify multiple of these axioms per query (averaged over positive and negative answers). Third, the metric #Q abstracts from the effort in terms of classified axioms and counts just the number of asked queries. In the light of these aspects it is clear that fewer normal queries suffice to gather the same information as obtained using a higher number of singletons. This tells us that normal queries are the best choice when #Q is the metric to be minimized.

Ad Q4: Drawing our attention to the lines in Figures 9 – 17, we clearly recognize that singleton queries require significantly less computation time than normal queries (regardless of the answering strategy[12]). In numerical terms, the savings in average computation time per query through the usage of singleton queries instead of normal ones amounts to between 80 % and 90 % over all ontologies. Please note, however, that absolute calculation times per query (stages 1 and 2, cf. Sec. 4.3) are very low (in the range of a few milliseconds) for both normal and singleton queries in all studied cases. Consequently, the time is definitely not a tie-breaker when deciding between both approaches. The justification for the computation speed-up when drawing on singleton queries as opposed to normal queries is the lower number of q-partitions that need to be explored in stage 1 of Algorithm 1 (cf. the "smaller search space" discussion in Sec. 4.2). This can be well read from Figures 9 – 17, where the red line (query computation time) changes proportionally to the black line (generated q-partitions).

Ad Q5: As the analyses and argumentations for Q1 – Q4 elucidate, singleton queries are by and large the best choice in case one would develop a debugging tool from scratch. The reasons for this conclusion in favor of singleton queries are—besides the pros enumerated in Sec. 4.2—their simplicity (interacting users need no advise whatsoever regarding the best answering strategy, etc.), their optimality and same performance achieved for all discussed expert types (all expert types coincide for singletons), their

[12] It may seem unnecessary to differentiate between different answering strategies when considering the query computation time. However, each answering behavior involves different numbers and types of test cases that are added upon a query's answer, and these can, in theory, affect the computation time of prospective queries.

time-efficiency (faster computation), and their superior performance in the majority of cases over normal queries (fewer required expert interactions for fault localization).

In case of already existing systems that draw on normal queries, experts should be advised to act according to the pragmatist answering strategy. In this case, an average performance comparable to singleton queries will be achieved.

6 Research Limitations and Future Work

The primary aim of this paper is to assess the usefulness of the new singleton query type for interactive fault localization in ontologies. As our results reveal, singleton queries indeed provide a reasonable and efficient means for expert consultation and, altogether, outperform existing interaction techniques. Thus, this work on the one hand testifies that fault localization using singleton queries is a promising topic for further research, and on the other hand provides first results in this direction.

However, this work also comes with limitations. First, our evaluations are based on simulations of debugging sessions and objective measures such as the number of required queries or classified axioms, or the computation times. Beside this objective assessment, of course, it is important to validate the subjective usability and acceptance of the approach, for instance in terms of a user study. This is part of our future work. However, we are nevertheless confident that users who are familiar using normal queries would likewise accept and adopt singleton queries. The first argument in this regard is that normal queries might be singletons as well, simply because they can contain *one* or more axioms. Second, there is no retraining or relearning whatsoever required to switch from the usage of normal queries to singletons, regardless of whether the expert is a query- or axiom-based type, since both querying approaches ask the user the same question, whether the set (or conjunction) of query-axioms is an entailment of the intended ontology. In fact, singleton queries even provide less space for misunderstandings and are easier explained to the user than normal ones as the implication of the *set of* axioms does not need to be clarified. Due to these points, we believe that the main results regarding the effectiveness of the query-based approach we obtained in our past user study [23] conducted for normal queries can be transferred to singletons as well.

A second limitation is the restriction to so-called explicit queries [20]—those that are constituted by axioms from the ontology at hand—in our theoretical and empirical analyses. The reason we did so is because we were able to devise an algorithm for the computation and optimization of explicit queries, by drawing on and extending the theory elaborated by [20]. The finding of an *efficient* algorithm that soundly generates implicit singleton queries, in contrast, is an open issue and on our future work agenda. As discussed in Sec. 3, this difficulty also explains why current approaches restrict themselves to normal (and not singleton) queries. Implicit queries are interesting particularly from the point of view of query complexity, i.e., how well an expert might understand (the axioms in) the query. A (syntax-based) model for estimating this complexity is suggested and evaluated in [23]. According to it, e.g., axioms like $A\ SubClassOf\ B$ are easier to comprehend for users if A, B are atomic classes rather than complex class expressions involving, e.g., negation or property restrictions. Whereas the syntactic (or structural) complexity of explicit queries depends on the complexity of (the axioms in)

the ontology, the shape of implicit queries can be controlled subject to the options offered by the used Description Logic reasoner [1]. Reasoners such as Pellet [38] or HermiT [36], for example, can be configured to restrict the computation of entailments to only specific axiom types, e.g., simple class subsumptions as mentioned above or basic class assertion axioms. In spite of this advantage over explicit queries, the use of *only* implicit queries leads to the loss of the guarantee [19, Prop. 7.5] that there is always a query to discriminate between two competing diagnoses. This underscores the importance of explicit queries, as discussed in this work.

As a third limitation, it should be noted that the analyzed expert types, as discussed in Sec. 3, provide by no means a complete characterization of all possible cases that could arise. While the discussion in this work bases on the assumption that an expert will provide for each query at the minimum as much information as is necessary to classify the entire query as a positive or negative test case (cf. the expert function in Sec. 2), there are (at least) two further query answering scenarios that are worthwhile considering. First, there is the case where the expert classifies a proper subset (or even none) of the axioms of a normal query positively while not labeling any axiom negatively, e.g., due to laziness or lack of knowledge. In such a scenario, the expert does not "implement" the expert function, as their answer leaves the classification of the query open—it could be negative if some of the remaining non-classified axioms is actually a non-entailment, or positive of all of them are entailments. Second, there is the case where an expert might misclassify axioms when answering queries. Such "oracle errors" were observed quite commonly in the studies conducted by [23]. Investigating these scenarios for normal and singleton queries as well as the conception of strategies how to handle these cases is another research avenue we will prospectively pursue.

In the light of these aspects, this work is just a first step towards understanding the impact of different interaction modes with users in the ontology fault localization domain. With the suggested singleton queries as an interface between expert and debugging system, however, it also gives a strategy that makes the overall fault finding process more efficient while not rendering the task more complicated.

7 Conclusions

We observe that existing approaches to query-based fault localization in ontologies interact with an expert by means of batch questions. That is, an expert is asked to classify *a set of* axioms as either a positive test case (the conjunction of axioms in the set is an entailment of the intended ontology) or as a negative one (some axiom is not an intended entailment). We point out that, on the one hand, there is a multitude of variants how an expert might answer such batch queries. In particular, we differentiate between four different expert types with regard to their query responses. On the other hand, current approaches ground the computation, selection and optimization of batch queries on the assumption of one particular of these answering behaviors. Since violations of this assumption turn optimizations into approximations and might lead to unexpected results and worse efficiency of the interactive fault localization process, we suggest as a remedy to use singleton queries, i.e., queries including exactly one axiom, to consult an expert. We elaborate a theory of computation and (heuristics-based) optimization

of singleton queries and provide complexity results for the suggested poly-time and poly-space algorithm.

Besides several apparent advantages of singleton queries in comparison to normal ones—such as a smaller search space, the facilitation of a precise (non-approximate) a-priori query assessment, or a more informative expert feedback—we conduct comprehensive empirical evaluations to gauge the usefulness of the new querying approach with regard to the expert waiting time between two queries and the effort necessary to locate the faulty axioms in the ontology. The main conclusions drawn from this study are:

1. Singleton queries are the overall best means of user consultation. The required expert interactions in terms of classified axioms are lower than for batch queries in the majority of diagnostic sessions, for almost all examined scenarios. Moreover, the time required for query computation and optimization is reduced by 80 to 90 % when using singletons. In absolute terms, it takes the proposed algorithm just a few milliseconds to obtain the heuristically-optimal query in the entire query search space. Furthermore, singleton queries are simpler and equally well suited for all different discussed query answering behaviors.
2. For batch queries, we find that there is a significant difference regarding the required expert interactions for fault localization for the various discussed query answering styles, with the best strategy being the chronological evaluation of axioms in the query until the first negatively classified one (if any) is found. In particular, this leads to less expert effort than classifying (i) all axioms per query or (ii) just a minimal subset of the query-axioms. When experts are properly advised to pursue the right answering strategy, then the costs of batch queries are comparable to singleton ones. This shows that both batch and singleton queries are, in general, reasonable approaches.

Finally, it is worthwhile noting that this approach is generally applicable for any knowledge representation language for which the entailment relation is monotonic (cf. [19]), e.g., Horn Logic, Propositional Logic, diverse constraint languages, as well as for other model-based diagnosis applications, as shown by [24].

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 1st edn. (2003)
2. Beck, K.: Test-driven development: by example. Addison-Wesley Professional (2003)
3. Ceusters, W., Smith, B., Goldberg, L.: A terminological and ontological analysis of the nci thesaurus. Methods of information in medicine **44**(4), 498 (2005)
4. Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artificial Intelligence **152**(2), 213–234 (2004)
5. Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Oberthaler, J., Parsia, B.: The national cancer institute's thesaurus and ontology. Journal of Web Semantics First Look 1_1_4 (2003)
6. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: Owl 2: The next step for owl. Web Semantics: Science, Services and Agents on the World Wide Web **6**(4), 309–322 (2008)

7. Horridge, M., Bail, S., Parsia, B., Sattler, U.: The cognitive complexity of owl justifications. In: International Semantic Web Conference. pp. 241–256. Springer (2011)
8. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is np-complete. Information processing letters **5**(1), 15–17 (1976)
9. Jannach, D., Schmitz, T., Shchekotykhin, K.: Parallel model-based diagnosis on multi-core computers. Journal of Artificial Intelligence Research **55**, 835–887 (2016)
10. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, University of Maryland, College Park (2006)
11. de Kleer, J., Raiman, O., Shirley, M.: One step lookahead is pretty good. In: Readings in model-based diagnosis. pp. 138–142. Morgan Kaufmann Publishers Inc. (1992)
12. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artificial Intelligence **32**(1), 97–130 (Apr 1987)
13. Meilicke, C.: Alignment incoherence in ontology matching. Ph.D. thesis, Universität Mannheim (2011)
14. Nikitina, N., Rudolph, S., Glimm, B.: Interactive ontology revision. J. Web Sem. **12**(0) (2012), http://www.websemanticsjournal.org/index.php/ps/article/view/233
15. Noy, N.F., Crubézy, M., Fergerson, R.W., Knublauch, H., Tu, S.W., Vendetti, J., Musen, M.A.: Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In: AMIA 2003 Open Source Expo. pp. 953–953 (2003)
16. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
17. Rector, A.L., Brandt, S., Schneider, T.: Getting the foot out of the pelvis: modeling problems affecting use of snomed ct hierarchies in practical applications. Journal of the American Medical Informatics Association **18**(4), 432–440 (2011)
18. Reiter, R.: A Theory of Diagnosis from First Principles. Artificial Intelligence **32**(1), 57–95 (1987)
19. Rodler, P.: Interactive Debugging of Knowledge Bases. Ph.D. thesis, Alpen-Adria Universität Klagenfurt (2015)
20. Rodler, P.: Towards better response times and higher-quality queries in interactive knowledge base debugging. Tech. rep., Alpen-Adria Universität Klagenfurt (2016), http://arxiv.org/pdf/1609.02584v2.pdf
21. Rodler, P.: On active learning strategies for sequential diagnosis. In: 28th International Workshop on Principles of Diagnosis (DX'17). vol. 4, pp. 264–283 (2018)
22. Rodler, P., Herold, M.: StaticHS: A variant of Reiter's hitting set tree for efficient sequential diagnosis. In: Proceedings of the Eleventh International Symposium on Combinatorial Search, SOCS 2018, Stockholm, Sweden - 14-15 July 2018. pp. 72–80 (2018)
23. Rodler, P., Jannach, D., Schekotihin, K., Fleiss, P.: Are Query-Based Ontology Debuggers Really Helping Knowledge Engineers? (2019), https://bit.ly/2TlxpFK
24. Rodler, P., Schekotihin, K.: Reducing model-based diagnosis to knowledge base debugging. In: 28th International Workshop on Principles of Diagnosis (DX'17). vol. 4, pp. 284–296 (2018)
25. Rodler, P., Schmid, W.: On the impact and proper use of heuristics in test-driven ontology debugging. In: Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18-21, 2018, Proceedings. pp. 164–184 (2018)
26. Rodler, P., Schmid, W., Schekotihin, K.: A generally applicable, highly scalable measurement computation and optimization approach to sequential model-based diagnosis. CoRR abs/1711.05508 (2017), http://arxiv.org/abs/1711.05508
27. Rodler, P., Schmid, W., Schekotihin, K.: Inexpensive cost-optimized measurement proposal for sequential model-based diagnosis. In: 28th International Workshop on Principles of Diagnosis (DX'17). vol. 4, pp. 200–218 (2018)
28. Rodler, P., Shchekotykhin, K., Fleiss, P., Friedrich, G.: RIO: Minimizing User Interaction in Ontology Debugging. In: Web Reasoning and Rule Systems, pp. 153–167 (2013)

29. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, (2016)
30. Schekotihin, K., Rodler, P., Schmid, W.: Ontodebug: Interactive ontology debugging plug-in for protégé. In: International Symposium on Foundations of Information and Knowledge Systems. pp. 340–359. Springer (2018)
31. Schekotihin, K., Rodler, P., Schmid, W., Horridge, M., Tudorache, T.: A protégé plug-in for test-driven ontology development. In: Proceedings of the 9th International Conference on Biological Ontology (ICBO 2018), Corvallis, Oregon, USA, August 7-10, 2018. (2018), http://ceur-ws.org/Vol-2285/ICBO_2018_paper_9.pdf
32. Schulz, S., Schober, D., Tudose, I., Stenzhorn, H.: The pitfalls of thesaurus ontologization–the case of the nci thesaurus. In: AMIA Annual Symposium Proceedings. vol. 2010, p. 727. American Medical Informatics Association (2010)
33. Shchekotykhin, K., Friedrich, G., Fleiss, P., Rodler, P.: Interactive Ontology Debugging: Two Query Strategies for Efficient Fault Localization. Web Semantics: Science, Services and Agents on the World Wide Web **12-13**, 88–103 (2012)
34. Shchekotykhin, K.M., Friedrich, G., Rodler, P., Fleiss, P.: Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In: ECAI. vol. 14, pp. 813–818 (2014)
35. Shchekotykhin, K.M., Jannach, D., Schmitz, T.: Mergexplain: Fast computation of multiple conflicts for diagnosis. In: IJCAI. vol. 15, pp. 3221–3228 (2015)
36. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient OWL reasoner. In: OWLED. CEUR Workshop Proceedings, vol. 432 (2008)
37. Siddiqi, S.A., Huang, J., et al.: Hierarchical diagnosis of multiple faults. In: IJCAI. pp. 581–586 (2007)
38. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)
39. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., et al.: The obo foundry: coordinated evolution of ontologies to support biomedical data integration. Nature biotechnology **25**(11), 1251 (2007)
40. Stuckenschmidt, H.: Debugging owl ontologies-a reality check. In: EON. vol. 359 (2008)
41. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting collaborative ontology development in protégé. In: International Semantic Web Conference. pp. 17–32. Springer (2008)

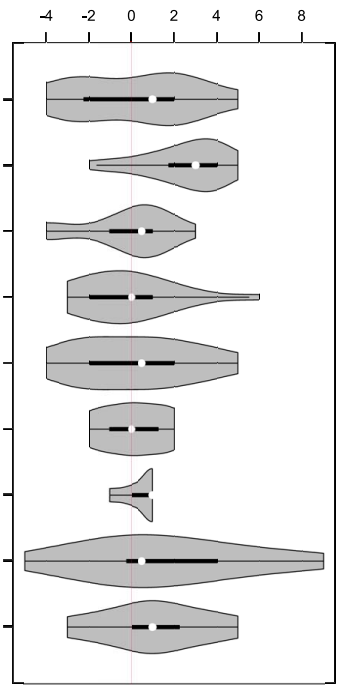

Fig. 2: ENT heuristic.

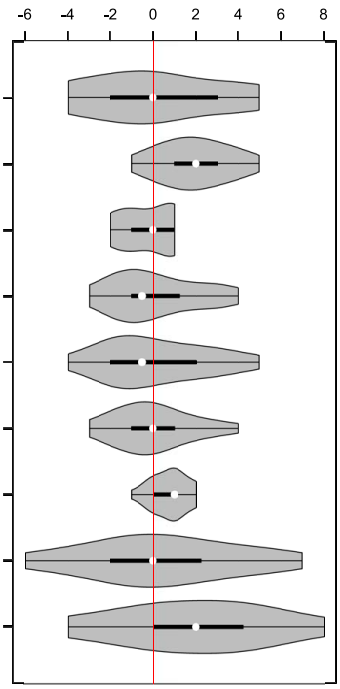

Fig. 3: SPL heuristic.

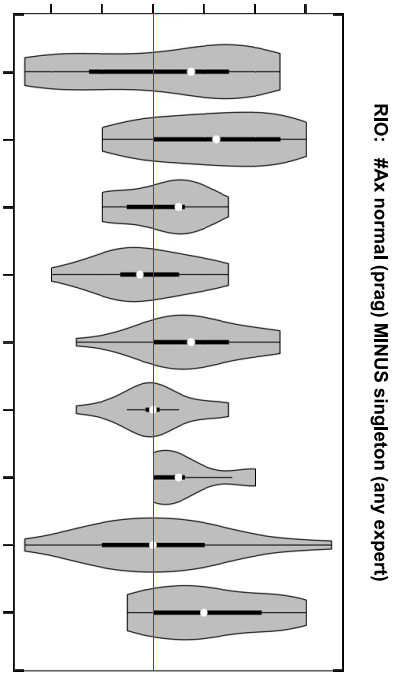

Fig. 4: RIO heuristic.

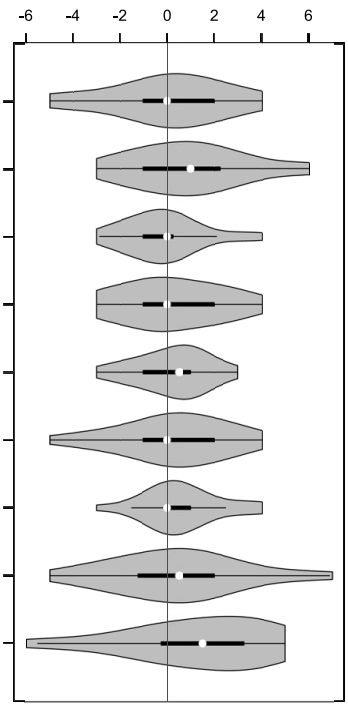

Fig. 5: RND heuristic.

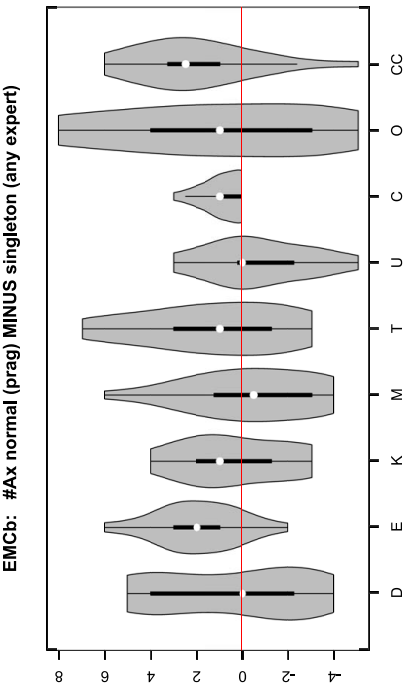

Fig. 8: EMCb heuristic.

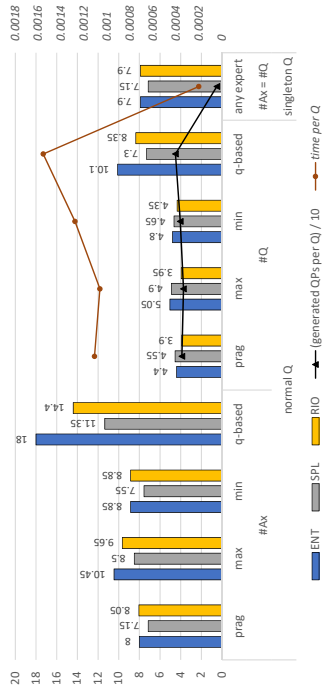

Fig. 9: U ontology overview

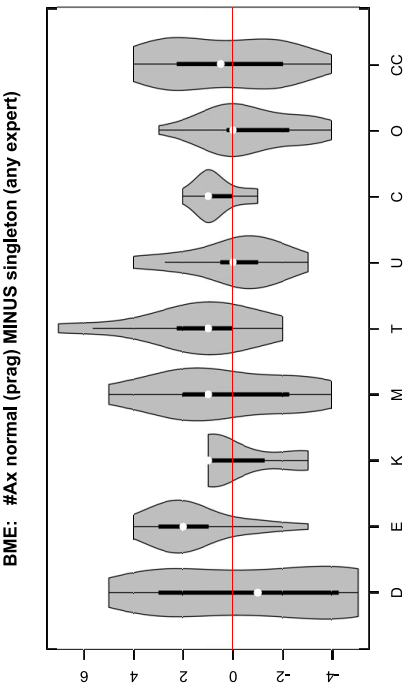

Fig. 6: BME heuristic.

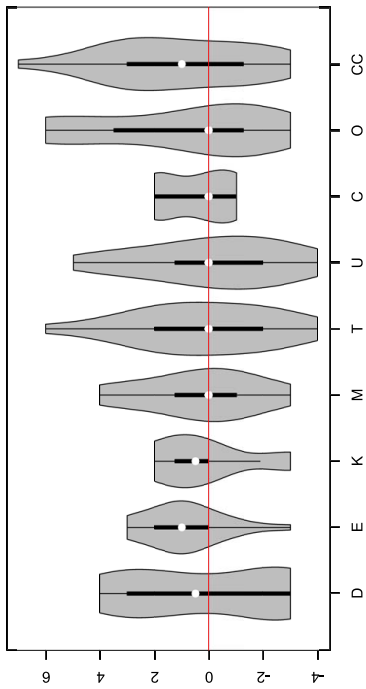

Fig. 7: KL heuristic.

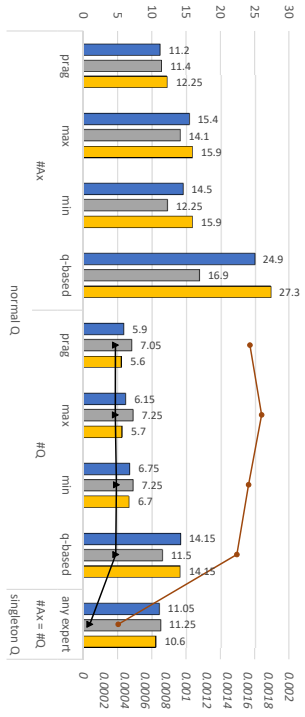

Fig. 10: T ontology overview.

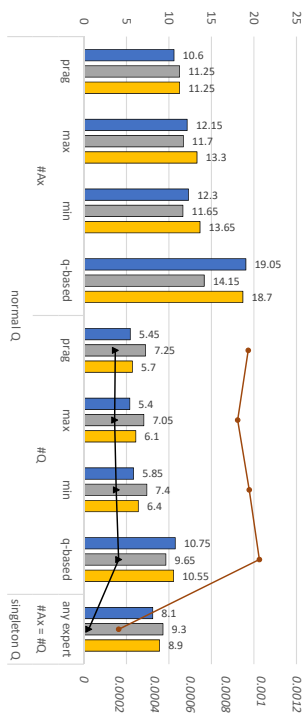

Fig. 12: E ontology overview.

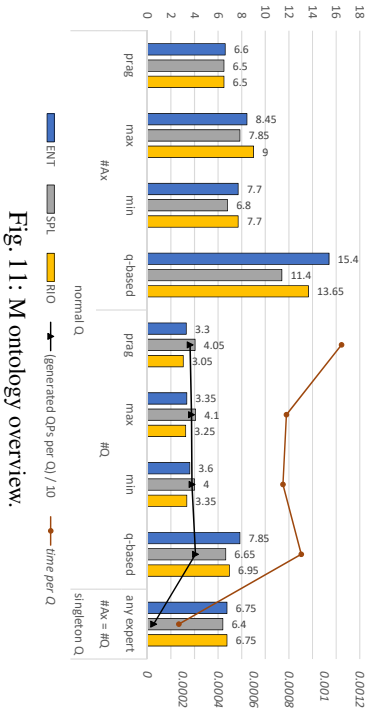

Fig. 11: M ontology overview.

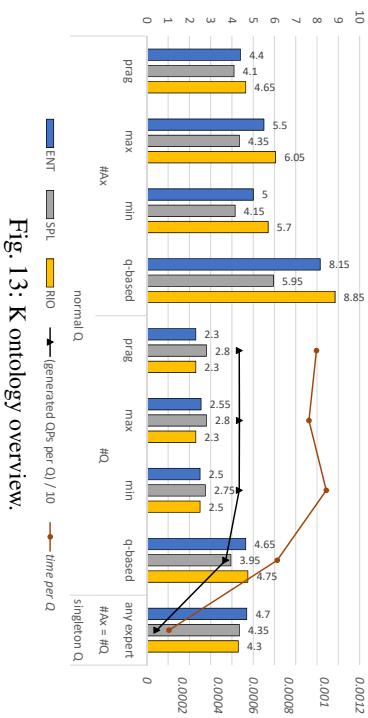

Fig. 13: K ontology overview.

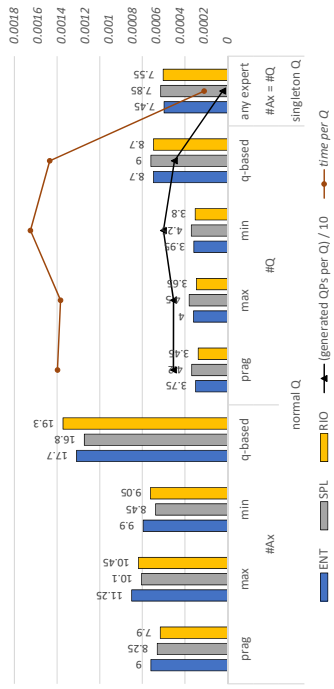

Fig. 16: O ontology overview.

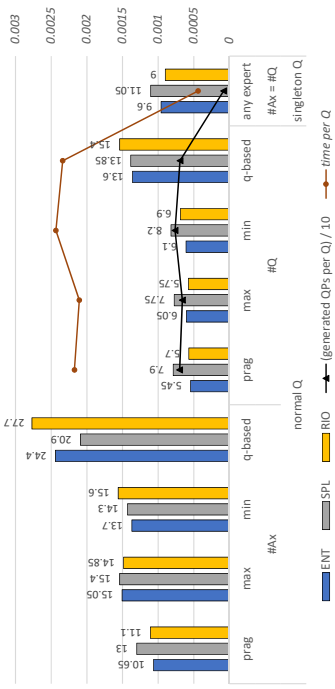

Fig. 17: CC ontology overview.

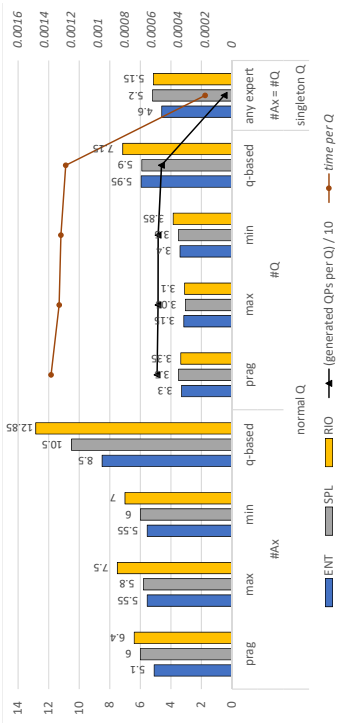

Fig. 14: C ontology overview.

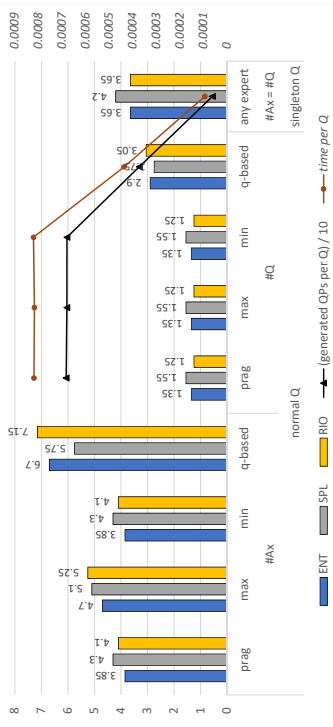

Fig. 15: D ontology overview.