Using Provenance to Debug Changing Ontologies [1]

Simon Schenk [a], Renata Dividino [a], Steffen Staab [a],

[a] *WeST - Institute for Web Science and Technologies*
University of Koblenz-Landau, Germany

Abstract

On the Semantic Web ontologies evolve and are managed in a distributed setting, e.g. in biomedical databases. Changes are contributed by multiple persons or organizations at various points in time. Often, changes differ by certainty or trustworthiness. When judging changes of automatically inferred knowledge and when debugging such evolving ontologies, the *provenance* of axioms (e.g. agent, trust degree and modification time) needs to be taken into account. Providing and reasoning with rich provenance data for expressive ontology languages, however, is a non-trivial task.

In this paper we propose a formalization of provenance, which allows for the computation of provenance for inferences and inconsistencies. It allows us to answer questions such as "When has this inconsistency been introduced and who is responsible for this change?" as well as "Can I trust this inference?".

We propose a black box algorithm for reasoning with provenance, which is based on general pinpointing, and an optimization, which enables the use of provenance for debugging in real time even for very large and expressive ontologies, such as used in biomedical portals.

Key words: Semantic Web, Provenance, Reasoning, Description Logics, OWL.

1. Introduction

Ontologies often evolve in open, multi user ontology editing environments. Examples of open, evolving ontologies are the Open Biomedical Ontologies repository of the US National Cancer Institutes Center for Bioinformatics [5] or the Ontology for Biomedical Investigations (`http://obi-ontology.org`), which is an integrated ontology for the description of life-science and clinical investigations. Many similar projects exist in varying domains. They are characterized by an open community with contributions at various points in time and from various sources, which might not be equally reliable.

In such open settings where conflicting changes can occur it is desirable to track two undesired situations:

(i) undesired inferences and
(ii) inconsistencies

In order to judge the reliability of inferences and to find errors in the ontology when debugging ontologies, it is hence necessary to ask questions such as "When has this inconsistency been introduced and who is responsible for this change?" as well as "Can I trust this inference?". *Provenance* information is used to answer these questions.

Email addresses: `sschenk@uni-koblenz.de` (Simon Schenk), `dividino@uni-koblenz.de` (Renata Dividino), `staab@uni-koblenz.de` (Steffen Staab).

[1] We would like to thank the US National Cancer Institute's Center for Bioinformatics for sharing the real world data used for the evaluation. This research was supported by the Federal Ministry of Education and Research of Germany under contract 01 IS 09 037 A-E, CollabCloud. The expressed content is the view of the authors but not necessarily the view of the CollabCloud project.

Provenance is *meta level information* and can be tracked in many dimensions: knowledge source, least recently modified dates, degrees of trustworthiness or the experience level of the editor. In all these cases, we have provenance labels, which are attached to axioms (e.g. timestamps) and orders over these labels, such as the ascending or descending order of timestamps or a partial order of trust among sources. This provenance information must be combined and propagated to a conclusion in order to answer the above mentioned questions.

Standard algorithms for debugging ontologies poorly support the user in answering meta level questions and require expensive reasoning. For these reasons they are not applicable for expressive and large scale real world ontologies.

With the approach presented in this paper we will show how to represent provenance and efficiently reason in OWL with provenance. Our approach helps the user cope with the complexity and dynamics of evolving ontologies.

1.1. *Debugging with Provenance*

Various approaches to the problem of debugging with provenance have been proposed. They can be grouped into three categories: (a) Extensions of given logical formalisms that deal with a particular type of provenance. Examples include extensions for debugging with uncertainty, such as fuzzy and probabilistic [24] or possibilistic [28] description logics. Recent proposals generalize such work to consider partial trust orderings [30]. (b) Flexible extensions for systems allowing for algebraic query evaluation (e.g. as relational databases and SPARQL engines), such as [10], [9] and [23], allow for many kinds of provenance, but are limited to lower expressiveness of the underlying logical formalism. (c) [33] provides a two step evaluation for provenance, which is very expressive, but which does not assign a uniform semantics to the definition and composition of provenance in the knowledge base.

Expressive descriptions of provenance combined with less expressive base languages (such as SPARQL and SQL) make use of the fact that the base languages can be evaluated bottom up using relatively simple algebraic expressions. However, debugging frameworks frequently have non-tree-based derivations used for consistency checking and querying. In order to be able to debug with algebraic provenance on top of such expressive base languages, we propose a debugging framework for provenance based on *pinpointing*. Pinpointing summarizes explanations for axioms in a single boolean formula, which then can be evaluated using a provenance algebra. Consequently, provenance for query answers can be computed by computing the explanations of the answer and using the pinpointing formula to compute provenance in an algebraic way.

Unfortunately, the computation of pinpoints may become very expensive and inapplicable if users need to interact with dynamically changing knowledge in real time. Therefore, we provide an optimized black-box algorithm, which does not need to compute *all* pinpoints. Our evaluation shows that our algorithm performs significantly better than the naïve algorithm, based on both real-world and synthetic datasets. For the purpose of this paper and for the prototypical implementation we restrict ourselves to a fragment of description logics and of OWL-2, called $\mathcal{SRIQ}(\mathcal{D})$, and use OWL-2 as a base language and consistency checks as debugging task. Other debugging tasks such as entailment checks and query answering can be reduced to consistency checks in OWL-2 [19].

1.2. *Contributions and Structure*

In this paper we motivate the need for tracking *provenance* when answering queries upon changing ontologies on the Semantic Web. We propose automatic mechanisms to verify the quality of a conclusion based on its available provenance and to compute provenance for inconsistencies. This paper makes the following contributions:

– a formalization of provenance syntax and semantics,
– the computation of provenance for inferences and inconsistencies based on pinpointing,
– an optimized algorithm for debugging with provenance, and
– an evaluation of the approach using real world data from evolving ontologies.

For the sake of example, we present our framework restricted to ontology diagnosis scenarios. Our work, however, introduces an approach for provenance querying under a variety of scenarios such as to restrictions of access rights[6], knowledge validity when the truth of knowledge changes with time[27], and inferring trust value[16].

The remainder of this paper is structured as follows. In Section 2 we formalize a use-case scenario to motivate the use of provenance for tracking ontologies changes. Section 3 introduces the foundations for provenance computation: first we briefly introduce an extension of the description logic DL, called $\mathcal{SRIQ}(\mathcal{D})$, underlying OWL lite and OWL DL, followed by the formalization of pinpointing and existing algorithms for pinpoint computations, and the formalization of provenance orders. Section 4 introduces the syntax used for expressing provenance in OWL-2. In Section 5 we define the semantics of provenance, including composition of provenance dimensions (to model complex provenance) and merging of conflicting provenance. In Section 6 we define the computation of provenance for a query answer based on pinpointing and propose an optimized algorithm for debugging with provenance. We discuss the complexity in Section 7. Section 8 presents the evaluation results. Our evaluation shows that this algorithm performs orders of magnitude better than a naïve implementation. A review of related work and the comparison with our approach is presented in Section 9. Section 10 concludes the paper.

2. Motivating Scenario

On 10th September 2009, a person called the German Press Agency (DPA) and notified them that a terrorist attack had just taken place in Bluewater, CA. In order to check the quality of this information, DPA did a short Internet research and found a website for Bluewater's local TV station (vpktv) and for the town itself. Additionally, they found Wikipedia entries for both the town and the local TV station. DPA announced the attack as breaking news.

Unfortunately for DPA, the information a was fake. No terrorist attack had happened in Bluewater and the town Bluewater, CA does not even exist. The breaking news had been spread and the fake background information had been set up by the marketing agency Neverest to support guerrilla marketing for the new movie "Shortcut to Hollywood".

DPA did not take into account the dynamics and the provenance of the information retrieved when checking the plausibility of the news. The new Wikipedia article has been created by the same author and modified multiple times shortly before the call. Moreover, all related websites as well as the Wikipedia articles had been set up by the same marketing agency at roughly the same time.

Table 1 shows the instantiation of our scenario. We suppose that the webpages of Bluewater and vpktv are described by axioms of ontologies from a single source, data from Wikipedia corresponds to an open, Wiki-like ontology editing system such as Freebase [2] , and the provenance consists of modification dates and respective degrees of trustworthiness based on the information source.

The axioms of Table 1 describe that a *RealCity* is a *City* with at least one *Broadcaster* (# 1) and a *Broadcaster* has its headquarters in a *City* (# 2). Additionally it defines that for any *Broadcaster* that has its headquarters in a *City* then this *City* *hasCompany* which is the *Broadcaster* (#3), that *bluewater* is a *City* (#4), *vpktv* is a *Broadcaster* (#5), and *vpktv* has its headquarters in the city *bluewater* (#6). Furthermore the columns *provenance* and *asserted* of Table 1 show provenance associated with each axiom. For instance, *DPA*, the German Press Agency, created axiom #1, *Nev*, the marketing agency Neverest, created axiom #4 and NevWP, their Wikipedia user, created axiom #5. The rightmost column indicates, when axioms have been asserted, for instance, the axiom #1 has been last asserted on 10th September 2009.

DPA needs to verify that Bluewater indeed is a city in CA. Hence, they need to answer the query "*bluewater: RealCity*?" and to obtain provenance for the answer. In the rest of this paper we will explain how this is done.

3. Foundations

Even though our approach is generic enough to be applicable beyond OWL, we focus on an extension of the description logic DL, called $\mathcal{SRIQ}(\mathcal{D})$, underlying OWL lite and OWL DL. Hence, we briefly revisit the definition of $\mathcal{SRIQ}(\mathcal{D})$.

As we use pinpointing as a vehicle for computing provenance, we then introduce pinpointing as a foundation for our algorithm for computing provenance and give an overview of algorithms for finding pinpoints.

For the scenario used in this paper we need two particular provenance dimensions: the *modification date* and *degrees of trustworthiness* which is derived from the knowledge source. In our scenario as well as in most other real world applications, *degrees of trustworthiness* can not result in a strict numeric trust order. Hence, we conclude the foundations by

[2] `http://freebase.org`

ID	axiom	source	date
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	DPA	2009-09-10
#2	$Broadcaster \sqsubseteq \exists hqIn.City$	DPA	2009-09-09
#3	$inverseProperty(hasCompany, hqIn)$	DPA	2009-09-10
#4	$bluewater : City$	Neverest	2009-09-09
#5	$vpktv : Broadcaster$	NeverestWikiP	2009-09-10
#6	$hqIn(vpktv, bluewater)$	NeverestWikiP	2009-09-09

Table 1
Ontology axioms describing our scenario and their provenance.

introducing generic, partial provenance orders as a generalization of various existing trust models.

3.1. $\mathcal{SRIQ}(\mathcal{D})$

In the following, we briefly introduce a fragment of DL, called $\mathcal{SRIQ}(\mathcal{D})$, and the fundamental reasoning problems related to DLs. For details we refer the reader to [19,17].

Definition 3.1 (Vocabulary) *A vocabulary $V = (N_C, N_R, N_I)$ is a triple where*
– *N_C is a set URIs used to denote classes,*
– *N_R is a set URIs used to denote roles and*
– *N_I is a set URIs used to denote individuals.*
N_C, N_R, N_I need not be disjoint.

An interpretation grounds the vocabulary in objects from an object domain.

Definition 3.2 (Interpretation) *Given a vocabulary V, an* interpretation *$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}_C}, \cdot^{\mathcal{I}_R}, \cdot^{\mathcal{I}_I})$ is a quadruple where*
– *$\Delta^{\mathcal{I}}$ is a nonempty set called the object domain;*
– *$\cdot^{\mathcal{I}_C}$ is the class interpretation function, which assigns to each class a subset of the object domain $\cdot^{\mathcal{I}_C} : N_C \to 2^{\Delta^{\mathcal{I}}}$*
– *$\cdot^{\mathcal{I}_R}$ is the role interpretation function, which assigns to each role a set of tuples over the object domain $\cdot^{\mathcal{I}_P} : N_P \to 2^{(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})}$;*
– *$\cdot^{\mathcal{I}_i}$ is the individual interpretation function, which assigns to each individual $a \in N_I$ an element $a^{\mathcal{I}_I}$ from $\Delta^{\mathcal{I}}$.*

Let $C, D \in N_C$, let $R, R_i, S \in N_R$ and $a, a_i, b \in N_I$. We extend the role interpretation function $\cdot^{\mathcal{I}_R}$ to role expressions:

$$(R^-)^{\mathcal{I}_R} = \{(\langle x, y\rangle) | (\langle y, x\rangle) \in R^{\mathcal{I}_R}\}$$

We extend the class interpretation function $\cdot^{\mathcal{I}_C}$ to class descriptions:

$$\begin{aligned}
\top^{\mathcal{I}_C} &\equiv \Delta^{\mathcal{I}}\\
\bot^{\mathcal{I}_C} &\equiv \emptyset\\
(C \sqcap D)^{\mathcal{I}_C} &\equiv C^{\mathcal{I}_C} \cap D^{\mathcal{I}_C}\\
(C \sqcup D)^{\mathcal{I}_C} &\equiv C^{\mathcal{I}_C} \cup D^{\mathcal{I}_C}\\
(\neg C)^{\mathcal{I}_C} &\equiv \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}_C}\\
(\forall R.C)^{\mathcal{I}_C} &\equiv \{x \in \Delta^{\mathcal{I}} | \langle x, y\rangle \in R^{\mathcal{I}_R} \to y \in C^{\mathcal{I}_C}\}\\
(\exists R.C)^{\mathcal{I}_C} &\equiv \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}_R}\}\\
(\exists R.Self)^{\mathcal{I}_C} &\equiv \{x \in \Delta^{\mathcal{I}} | \langle a, a\rangle \in R^{\mathcal{I}}\}\\
(\geq nR)^{\mathcal{I}_C} &\equiv \{x \in \Delta^{\mathcal{I}} | \exists y_1, ..., y_m \in \Delta^{\mathcal{I}} :\\
&\quad \langle x, y_1\rangle, ..., \langle x, y_m\rangle \in R^{\mathcal{I}_R} \wedge m \geq n\}\\
(\leq nR)^{\mathcal{I}_C} &\equiv \{x \in \Delta^{\mathcal{I}} | \nexists y_1, ..., y_m \in \Delta^{\mathcal{I}} :\\
&\quad \langle x, y_1\rangle, ..., \langle x, y_m\rangle \in R^{\mathcal{I}_R} \wedge m > n\}
\end{aligned}$$

Class expressions are used in axioms.

Definition 3.3 (Axiom) *An axiom is one of*
– *A general concept inclusion of the form $C \sqsubseteq D$ for concepts C and D;*
– *An individual assertion of one of the forms $a : C$, $(a, b) : R$, $(a, b) : \neg R$, $a = b$ or $a \neq b$ for individuals a, b and a role R.*
– *A role assertion of one of the forms $R \sqsubseteq S$, $R_1 \circ ... \circ R_n \sqsubseteq S$, $Asy(R)$, $Ref(R)$, $Irr(R)$, $Dis(R, S)$ for roles R, R_i, S.*

We now define satisfaction of axioms.

Definition 3.4 (Satisfaction of Axioms) *Satisfaction of axioms in an interpretation $\mathcal{I}$ is defined as follows. With $\circ$ we denote the composition of binary relations.*

$$\begin{aligned}
(R \sqsubseteq S)^{\mathcal{I}} &\equiv \langle x, y\rangle \in R^{\mathcal{I}} \to \langle x, y\rangle \in S^{\mathcal{I}}\\
(R_1 \circ ... \circ R_n \sqsubseteq S)^{\mathcal{I}} &\equiv \forall \langle x, y_1\rangle \in R_1^{\mathcal{I}}, \langle y_1, y_2\rangle \in R_2^{\mathcal{I}}, ...,\\
&\quad \langle y_{n-1}, z\rangle \in R_n^{\mathcal{I}} : \langle x, z\rangle \in S^{\mathcal{I}}
\end{aligned}$$

$$
\begin{aligned}
(Asy(R))^{\mathcal{I}} &\equiv \exists \langle x,y\rangle \in R^{\mathcal{I}} : \langle y,x\rangle \notin R^{\mathcal{I}}\\
(Ref(R))^{\mathcal{I}} &\equiv \forall x \in \Delta^{\mathcal{I}} : \langle x,x\rangle \in R^{\mathcal{I}}\\
(Dis(R,S))^{\mathcal{I}} &\equiv R^{\mathcal{I}} \cap S^{\mathcal{I}} = \varnothing\\
(Irr(R))^{\mathcal{I}} &\equiv \forall x \in \Delta^{\mathcal{I}} : \langle x,x\rangle \notin R^{\mathcal{I}}\\
(C \sqsubseteq D)^{\mathcal{I}} &\equiv x \in C^{\mathcal{I}} \rightarrow x \in D^{\mathcal{I}}\\
(a : C)^{\mathcal{I}} &\equiv a^{\mathcal{I}} \in C^{\mathcal{I}}\\
((a,b) : R)^{\mathcal{I}} &\equiv \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\\
((a,b) : \neg R)^{\mathcal{I}} &\equiv \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \notin R^{\mathcal{I}}\\
a = b &\equiv a^{\mathcal{I}} = b^{\mathcal{I}}\\
a \neq b &\equiv a^{\mathcal{I}} \neq b^{\mathcal{I}}.
\end{aligned}
$$

An ontology is comprised of a set of axioms.

Definition 3.5 (Ontology) *A $\mathcal{SRIQ}(\mathcal{D})$ ontology O is a set of axioms as defined in definition 3.3.*

In this paper we make use of two basic reasoning mechanisms for $\mathcal{SRIQ}(\mathcal{D})$, i.e. consistency checking and axiom entailment.

Definition 3.6 (Reasoning with $\mathcal{SRIQ}(\mathcal{D})$) *We say an interpretation $\mathcal{I}$ of an ontology O is a model of O ($\mathcal{I} \models O$), if all axioms in O are satisfied in I.*

We say an ontology O is consistent if there exists a model of O.

We say an axiom A is entailed by an ontology O ($O \models A$), if A is satisfiable in all models of O.

Obviously, a subontology again is an ontology. Moreover, if $O' \subseteq O$ and $O' \models A$, then also $O \models A$. We will make use of this fact in the optimized algorithm proposed in Section 6.2.

3.2. *Pinpointing*

The term pinpointing has been coined for the process of finding explanations for concluded axioms or for a discovered inconsistency. A pinpoint is a minimal subset of an ontology, which makes the concluded axiom true (or the theory inconsistent, respectively). Such an explanation is called a pinpoint. While there may be multiple ways to establish the truth or falsity of an axiom, a pinpoint describes exactly one such way.

Definition 3.7 (Pinpoint) *A pinpoint P for an axiom A wrt. an ontology O is a set of axioms, such that $P \subseteq O$, $P \models A$, and $\forall B \in P : P \setminus \{B\} \not\models A$.*

Analogously, we can define a justification for a refuted axiom ($O \models \neg A$) as $P \subseteq O$, $P \models \neg A$, and $\forall B \in P : P \setminus \{B\} \not\models \neg A$. Hence, finding pinpoints for a refuted axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [1]. In this paper we will focus on entailed axioms ($O \models A$). However, all definitions and algorithms can be modified to justifications as well.

Pinpointing is the computation of all pinpoints for a given axiom A and ontology O. The *pinpointing formula* [7] describes, which axioms need to be present to for O to entail A.

Definition 3.8 (Pinpointing Formula) *Let A be an axiom, O an ontology and $P_1, ..., P_n$ with $P_i = A_{i,1}, ..., A_{i,m_i}\}$ the pinpoints of A wrt. O. Let* id *be a function assigning a unique identifier to an axiom. Then $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} id(A_{i,j})$ is a pinpointing formula of A wrt. O.*

Algorithms for finding pinpoints can be grouped into three groups:

Finding one Pinpoint Algorithms for finding one pinpoint can either derive a pinpoint by tracking the reasoning process of a tableaux reasoner, or use an existing reasoner as a black box. In the latter case, a pinpoint is searched for by subsequently growing (shrinking) a subontology, until it starts (stops) entailing the axiom under a pinpoint is searched for. Based on the derived smaller ontology the process is refined, until a pinpoint has been found. The advantage of blackbox algorithms is that they can support any DL, for which a reasoner is available [1].

Finding all Pinpoints using a Tableaux Reasoner Baader and Peñaloza have shown how tableaux reasoners for DLs such as OWL can be extended to find pinpointing formulas [7]. In their approach a tableaux reasoner is extended to find not only one, but all pinpoints. Special care needs to be taken in order to ensure termination of the tableaux algorithm. As an advantage, the overhead for pinpointing is lower compared to a blackbox algorithm. Moreover, this approach can derive a compact representation of the pinpointing formula, which however might still have worst-case exponential size in normal form.

Finding all Pinpoints using Blackbox Algorithms The most performant black-box algorithms extract a relevant module from the overall ontology, ensuring that this module yields the same inferences with respect to the axiom in interest. Then, starting from a single pinpoint, Reiter's Hitting Set Tree algorithm [29] is used to compute all pinpoints by iteratively removing one axiom from the pinpoint at hand and growing it to a full pinpoint again [21,20].

3.3. *Provenance Order*

Provenance may come in various, complex dimensions such as knowledge source, editors, modification data and degrees of trustworthiness. Provenance dimensions must be exploited and combined to arrive at an accurate assessment of information value [15]. Provenance dimensions are defined in detail in Section 5.1.

First we take a closer look at the two specific dimensions of provenance used in examples in this paper: time and source. Provenance is expressed using *provenance labels*. An example for a provenance label is a timestamp or a source name.

The label alone is not sufficient for the tracking of provenance when debugging an ontology. We need a *provenance order* on the provenance labels. For example, suppose that an user has introduced an inconsistency during his last change. We might be interested, when the oldest axiom leading to an inconsistency has been added, or when the youngest has been added. The oldest axiom tells us, when this particular topic has first been addressed and the newest one tells us, when the inconsistency has been introduced. We use an ascending or descending order of timestamps.

Not all types of provenance labels have a natural order. There are, for example, many ways how *degrees of trustworthiness* can be computed. Often simplifications are used, such as assuming trust to be measured on a scale from 1 to 10. Such simplifications usually are not easily justifiable [15] when trust should be established using provenance labels such as the knowledge source.

In particular, trust (and provenance in general) can not always be measured on a total order, but there may be agents which are incomparable. Please note that even though we use trust in our running example, the same applies e.g. when modeling access rights, roles in an editing workflow or comparing world views of users participating in an ontology editing platform. A good analogy are access right systems, which usually introduce some kind of ordering of users and groups. This ordering always is artificial and usually also partial (not in every pair of users or groups one is more powerful than the other) [6].

We provide a generic formalization of provenance orders here, which subsumes others such as [13] and enables us to use any kind of order over provenance labels in the following. The following formalization has been used in similar form in [30].

source	trust
DPA	∞
Neverest	Nev
NeverestWP	NevWP

Table 2
Correspondences between source and degrees of trustworthiness.

Definition 3.9 (Provenance Order) *A provenance order $\mathcal{T}$ is a lattice over a finite set of* provenance labels. ∞ *is the label used for the maximal element of the lattice.*

If two provenance labels a and b are not comparable, we introduce virtual provenance label inf_{ab} and sup_{ab}, such that:
– $inf_{ab} < a < sup_{ab}$ and $inf_{ab} < b < sup_{ab}$;
– $\forall_{c<a,c<b} : c < inf_{ab}$ and
– $\forall_{d>a,d>b} : d > sup_{ab}$

To understand the importance of the last two steps, assume that $c > a > d$ and $c > b > d$ and a, b are incomparable. Then the provenance label of $a \wedge b$ would be the provenance label of c, as c is the supremum in the provenance order. Obviously this escaping to a higher provenance label is not desirable. In our trust example, it would mean escaping to a higher trust level. Considering roles is a workflow, we might end up with the wrong role. Instead, the virtual provenance labels represent that we need to pick at least one, a or b. For convenience, we will write $\{a, b\}$ for $inf_{a,b}$ in the following.

Note that in practice, these values need not be precomputed, so exponential blowup is not an issue. If they are needed they can be encoded as lists of atomic labels.

Provenance orders subsume strict orders (such as [0..1], `xsd:datetime` and numeric trust degrees computed for Wikipedia [2]). It also allow for incomparable provenance labels, which are common on the Web due to the sheer size and usually incomplete knowledge.

In this paper we assign degrees of trustworthiness to axioms based on the user expertise by whom they have been modified. Hence, the *knowledge source* of a piece of information is used to establish *trust*.

In our running example, we use the following correspondences between source and degrees of trustworthiness presented in Table 2.

Figure 1 illustrates a trust order for the sources shown in Table 1 from the perspective of DPA. ∞ is assigned to the most trustworthy source which is

DPA and *Nev, NevWP* are incomparable.

$$\begin{tikzcd}
& \infty \arrow[d, dash] & \\
& \mathrm{sup}_{\mathrm{Nev,NevWP}} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathrm{Nev} \arrow[dr, dash] & & \mathrm{NevWP} \arrow[dl, dash] \\
& \mathrm{inf}_{\mathrm{Nev,NevWP}} \arrow[d, dash] & \\
& \bot &
\end{tikzcd}$$

Fig. 1. Provenance order of the knowledge sources.

4. Syntax of Provenance

Provenance in OWL-2 can be expressed as annotations on axioms. Annotations are of importance for the management of ontologies as annotations may be used to support analysis during collaborative engineering. Basically, an axiom annotation assigns an annotation object to an axiom. For instance in our scenario the axiom "$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$" is assigned the information source annotation object *DPA*.

A provenance annotation consists of an annotation URI and a provenance object specifying the value of the annotation. In our case, the provenance object is a constant-value representing who asserted the axiom, when the axiom was last asserted, the degree of trustworthiness of the axiom, or a combination thereof. We provide a detailed grammar for provenance annotations in [31]. The grammar for provenance annotations as an extension of OWL-2 annotations [3] is as follows. For sake of clarity we use the prefix *Provenance* for extensions to the OWL-2 grammar, which uses prefix *OWL*.

OWLAnnotation := **ProvenanceAxiomAnnotation**
ProvenanceAxiomAnnotation := 'ProvenanceAxiomAnnotation' '(' **IRI ProvenanceAnnotation**$^{+}$ ')'
ProvenanceAnnotation := **ProvenanceCertaintyAnnotation** | **ProvenanceDateAnnotation** | **ProvenanceSourceAnnotation**
ProvenanceCertaintyAnnotation := 'ProvenanceCertaintyAnnotation' '(' **Value** ')'

[3] OWL 2 Web Ontology Language: Spec. and Func.-Style Syntax: http://www.w3.org/TR/2008/WD-owl2-syntax-20081202

ProvenanceSourceAnnotation := 'ProvenanceSourceAnnotation' '(' **Value** ')'
ProvenanceDateAnnotation := 'ProvenanceDateAnnotation' '(' **Value** ')'

An example of how provenance is represented and associated with OWL axioms is presented below. We consider the axioms #4 and #5 from our scenario:

```
OWLAxiomAnnotation(ClassAssertion(bluewater City)
 ProvenanceAxiomAnnotation(
   annot1 ProvenanceSourceAnnotation(Nev)))
OWLAxiomAnnotation(
 ObjectPropertyAssertion(hqIn vpktv bluewater)
 ProvenanceAxiomAnnotation(
   annot2 ProvenanceDateAnnotation(09.09.2009)))
```

5. Semantics of Provenance

Provenance assignments are syntactically expressed in OWL-2 using axiom annotations. Annotations, however, have no semantic meaning in OWL-2. All annotations are ignored by the reasoner, and they may not themselves be structured by further axioms.

Furthermore, using such an abstract syntax may remain remarkably ambiguous if it cannot be linked to a formal semantics. Assume that the following provenance axioms are part of our scenario:

ID	axiom	trust
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	∞
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	Nev

Table 3
Example of multiple provenance assigments to the same axiom.

For the same axiom identified by #1 presented in Table 5, the question may arise whether this means a disjunction, i.e. one of the two sources has provided the fact, or a conjunction, i.e. both sources have provided the fact, or a collective reading, i.e. the two sources together gave rise to the fact, or whether this situation constitutes invalid provenance. In order to prevent such ambiguities we introduce a generic semantic framework for provenance.

5.1. *Provenance Dimension*

Provenance can have atomic and complex dimensions such as knowledge source, modification date or a composition of thereof. We assume that these (and possible further) dimensions are independent of each other. In the next section, we generalize from this assumption.

Definition 5.1 (Provenance Dimension) *A provenance dimension D is an algebraic structure $(B_D, \dot{\vee}_D, \dot{\wedge}_D)$, such that $(B_D, \dot{\vee}_D)$ and $(B_D, \dot{\wedge}_D)$ are complete semilattices. We denote the minimal element of D by $\bot_D$.*

B_D represents the labels the provenance can take, e.g. all valid timestamps for the modification date. As $(B_D, \dot{\vee}_D)$ and $(B_D, \dot{\wedge}_D)$ are *complete* semilattices, they are, in fact, also lattices. Hence, a finite set of provenance labels always has a join (supremum, least upper bound) and a meet (infimum, greatest lower bound) wrt. the corresponding order.

In contrast to provenance orders defined in Section 3.3 the join and meet operators in a provenance dimension need not be dual as they can come from two different lattices, which share the same values but have different orders. An example of a provenance dimension, where this may become important is *where provenance* [9]: Only track, who contributed in any way to a certain inference. In this case, join and meet would coincide and would both be set unions of information sources. In all dimensions discussed in this paper, join and meet are dual. In this case, provenance dimension and provenance order directly correlate.

Example 5.1 *To illustrate the meaning of $\dot{\wedge}$ and $\dot{\vee}$, let I be the provenance interpretation that is a partial function mapping axioms into the label range of trust, and A and B be axioms of an ontology such that $A \neq B$. When combining two provenance labels from D, which are assigned to A and B, the intuitive meaning of $\dot{\vee}$ is "I need to trust one of A and B" (corresponding to a logical "or"). The intuitive meaning of $\dot{\wedge}$ is "I need to trust both A and B" (corresponding to a logical "and")). Hence, trust can be modeled as:*
– $I_{trust}(A \dot{\vee} B) = sup(I_{trust}(A), I_{trust}(B))$
– $I_{trust}(A \dot{\wedge} B) = inf(I_{trust}(A), I_{trust}(B)$

Likewise, the modification date as well as the creation date can be modeled as:
– $I_{date}(A \dot{\vee} B) = min(I_{date}(A), I_{date}(B))$
– $I_{date}(A \dot{\wedge} B) = max(I_{date}(A), I_{date}(B))$

As we have seen in the Example 5.1, provenance is assigned to ontology axioms. Within a single assignment, the provenance must be uniquely defined.

Definition 5.2 (Provenance Assignment) *A provenance assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of a provenance dimension and a corresponding provenance label, such that $D_i = D_j \Rightarrow d_i = d_j$. As default label for a provenance assignment in dimension D_j we choose the minimal element $\bot_{D_j}$. By $ProvAss(A)$ we denote the set of provenance assignments for an axiom A.*

Example 5.2 *As an example, for the axiom* "bluewater: City" *of our running example, we have the following provenance assignment:*

$$ProvAss(bluewater : City) = \{(trust, Nev), (date, 2009-09-09)\}$$

Without loss of generality we assume a fixed number of provenance dimensions.

Next, we formalize how provenance assignments are composed. To obtain a logical formula which express how provenance assignments are composed, called *provenance formula*, we make use of pinpointing formulas (discussed in Section 3.2) and of the *how provenance* strategy. *How provenance* [14] is a strategy, which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a boolean formula connecting the A_i. As pinpointing summarizes explanations for axioms in a single boolean formula, and thus it provides *how provenance*, we use it to come up with a provenance formula.

Example 5.3 *Consider the query "for each city, find all companies located in that city":*

$$x{:}City \wedge hasCompany(x,y).$$

The result of this query and the corresponding pinpointing formula are, based on the example data from Table 1:

x	*y*	*pinpointing formula*
bluewater	*vpktv*	#3∧#4∧#6

The associated provenance formula for this query result is:

provenance formula
ProvAss*(#3)* $\dot{\wedge}$ ProvAss*(#4)* $\dot{\wedge}$ ProvAss*(#6)*

To evaluate the corresponding provenance formula, we need to define the operators for provenance dimensions.

Definition 5.3 (Provenance Operations) *Let A, B be axioms. Let $ProvAss(A) = \{(D_1, x_1), ..., (D_n, x_n)\}$ and $ProvAss(B) = \{(D_1, y_1), ..., (D_n, y_n)\}$ be provenance assignments. Then the provenance operations $\dot{\vee}$ and $\dot{\wedge}$ are defined as follows:*

$$ProvAss(A) \dot{\vee} ProvAss(B) = \{(D, x \dot{\vee}_D y) | (D, x) \in ProvAss(A) \text{ and } (D, y) \in ProvAss(B)\}.$$

$$ProvAss(A) \dot{\wedge} ProvAss(B) = \{(D, x \dot{\wedge}_D y) | (D, x) \in ProvAss(A) \text{ and } (D, y) \in ProvAss(B)\}.$$

Example 5.4 *To illustrate how provenance is derived, for instance, the provenance assignment for the axiom* "bluewater: City" *is* {(trust, Nev), (date, 2009-09-09)}*, and the provenance assignment for the axiom* "hqIn (vpktv,bluewater)" *it is* {(trust, NeverestWikiP), 2009-09-09)}*. We assume the provenance order described in Table 2. The provenance for* bluewater: City ∨ hqIn(vpktv, bluewater) *is determined as follows:*

$$\begin{aligned}&ProvAss(bluewater : City) \dot{\vee} \\ &\quad ProvAss(hqIn(vpktv, bluewater)) = \\ &\qquad \{(trust, Nev \dot{\vee} NevWP), \\ &\qquad (date, 2009\text{-}09\text{-}09 \dot{\vee} 2009\text{-}09\text{-}09)\}.\end{aligned}$$

Note that, due to the defaults introduced in Definition 5.2, the operations on provenance assignments are defined even in the presence of incomplete provenance from a domain. In our framework, just as $\dot{\vee}$ and $\dot{\wedge}$ correspond to a logical "or" and "and", default corresponds to a default truth value of "unknown" in a default logic. With a default assignment, we provide a uniform treatment of axioms in case of absence of provenance, and thus we are able to combine arbitrarily provenance assignments. In any case, as illustrated in Example 5.1, the interpretation of provenance and defaults assignments as well as the interpretation of the provenance operations are domain application dependent. Such considerations are orthogonal to our framework.

While axioms in the underlying description logic may contain negation, these negations are not visible and not needed at the level of the provenance. Hence, the provenance algebra does not need to contain a negation operator.

Finally, we define how to retrieve the provenance assigned to an axiom A within a provenance dimension. The provenance of an axiom A within a provenance dimension is obtained by evaluating the corresponding provenance formula in the dimension under consideration.

Definition 5.4 (Provenance Evaluation) *Let prov(A) be a function mapping from an axiom A to a provenance assignment in dimension D. The provenance of an axiom A wrt. O in D is obtained by computing a pinpointing formula ϕ of A wrt. O and obtaining ψ by replacing each axiom in ϕ with its provenance assignment in D and the logical operators $\dot{\vee}$ and $\dot{\wedge}$ with their corresponding operators in D. Then $prov(A)$ is computed by evaluating ψ.*

Example 5.5 *In our running example, for the query "for each city find all companies",* x:City ∧ hasCompany(x,y)*, we have the result* bluewater, vpktv *and the following provenance formula:*

$$ProvAss(\#3) \dot{\wedge} ProvAss(\#4) \dot{\wedge} ProvAss(\#6)$$

The corresponding provenance evaluation for the dimension trust is:

$$\begin{aligned}&(trust, \infty) \dot{\wedge} (trust, Nev) \dot{\wedge} (trust, NevWP) = \\ &\qquad (trust, \infty \dot{\wedge} Nev \dot{\wedge} NevWP) = \\ &\qquad\qquad (trust, inf_{Nev, NevWP}).\end{aligned}$$

5.2. *Complex Provenance Dimensions*

In the previous section we have described, how provenance can be computed in a single dimension. This focus on a single dimension is useful if independence of dimensions can be assumed. Sometimes, however, this is not the case. For example when a group of users collaboratively edits an ontology, the time an axiom was asserted and the user responsible for the modification will often correlate. In this case, two knowledge dimensions can be composed into a complex provenance dimension:

Definition 5.5 (Composition of Dimensions) *Let $D_1 = (B_{D_1}, \dot{\vee}_{D_1}, \dot{\wedge}_{D_1})$ and $D_2 = (B_{D_2}, \dot{\vee}_{D_2}, \dot{\wedge}_{D_2})$ be provenance dimensions. Then $D = (B_D, \dot{\vee}_D, \dot{\wedge}_D)$ is a composed provenance dimension, such that*
- $B_D = B_{D_1} \times B_{D_2}$,
- $(B_D, \dot{\vee}_D) = \{((x, y)(v, w)) | x, v \in B_{D_1}$ *and* $y, w \in B_{D_2}$ *and* $x \leq_{\vee_{D_1}} v$ *and* $y \leq_{\vee_{D_2}} w\}$, *and*
- $(B_D, \dot{\wedge}_D) = \{((x, y), (v, w)) | x, v \in B_{D_1}$ *and* $y, w \in B_{D_2}$ *and* $x \leq_{\wedge_{D_1}} v$ *and* $y \leq_{\wedge_{D_2}} w\}$

We will refer to the elements of B_D as complex provenance labels *and to the elements of B_{D_1} and B_{D_2} as* atomic provenance labels*.*

Example 5.6 *As an example of composition of dimensions, we compose the dimensions trust and modification date and compute the provenance for* x:City ∧ hqIn(y,x)*. We have the provenance formula:*

$$ProvAss(\#4)\dot{\wedge}ProvAss(\#6)$$

If we treat the trust and modification date dimensions separately, the results are

$$\{(trust, Nev))\}\dot{\wedge}\{(trust, NevWP)\} = \{(trust, inf_{Nev,NevWP})\}$$

$$\{(date, 2009\text{-}09\text{-}09)\}\dot{\wedge}\{(date, 2009\text{-}09\text{-}10)\} = \{(date, 2009\text{-}09\text{-}10)\}$$

If we combine them into one dimension as shown in Figure 2, however, the result is

$$\{(trust, Nev), (date, 2009\text{-}09\text{-}09)\}\dot{\wedge} \{(trust, NevWP), (date, 2009\text{-}09\text{-}10)\} = \{(trust, NevWP), (date, 2009\text{-}09\text{-}10)\}$$

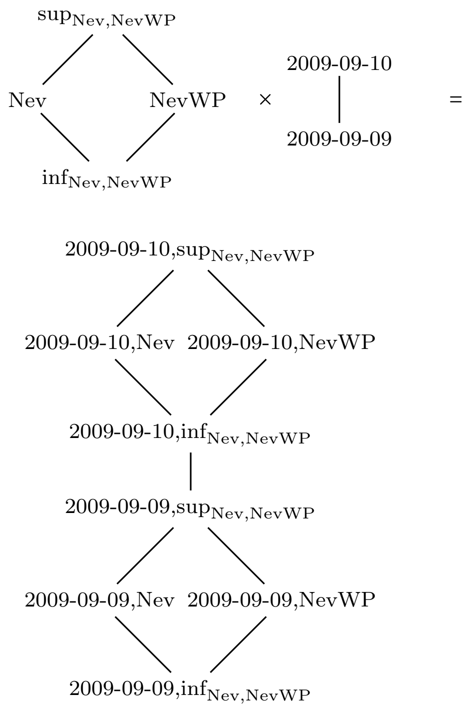

Fig. 2. Provenance order of the knowledge sources.

Having composed the interdependent dimensions into one, one may use the composed provenance dimension just as an atomic one. Definition 5.4 applies exactly as for simple provenance dimensions.

5.3. *Semantics for Conflicting Provenance*

In the following we extend our model to support conflicting provenance, which can arise from conflicting changes or provenance assignments by multiple users at different times. A new operator is needed for this merging, as the merge operator needs not coincide with one of $\dot{\vee}$ and $\dot{\wedge}$.

Definition 5.6 (Conflict Tolerant Dimension) *A conflict tolerant provenance dimension D is an algebraic structure $(B_D, \dot{\vee}_D, \dot{\wedge}_D, \oplus_D)$, such that $(B_D, \dot{\vee}_D)$, $(B_D, \dot{\wedge}_D)$ and $(B_D, \oplus_D)$ are complete semilattices. The minimum of $(B_D, \oplus_D)$ is $\perp_D$.*

Example 5.7 *To illustrate how to support conflicting provenance by multiple provenance assignments, let I be a provenance interpretation and let A be an axiom of an ontology, for which there exist multiple provenance assignments. We use A' and A'' to represent the axiom A with different provenance assignments. Let us compare the already known provenance dimension* modification date *with the* creation date*. For* modification date*, $\oplus$ needs to be the* maximum *(we are interested in the last assertion), while for* creation date*, $\oplus$ needs to be the* minimum *(we are interested in the first assertion). In contrast to example 5.1 the operators for creation date and modification date do not coincide. The* modification date *could be modeled as:*

$$I_{date}(A' \oplus A'') = max(I_{date}(A'), I_{date}(A''))$$

The creation date *could be modeled as:*

$$I_{cre}(A' \oplus A'') = min(I_{cre}(A'), I_{cre}(A''))$$

Likewise, the trust *could be modeled as:*

$$I_{trust}(A' \oplus A'') = sup(I_{trust}(A'), I_{trust}(A''))$$

To show how the support to conflicting provenance by multiple provenance assignments can be applied to our scenario, we slightly extend our running example in Example 5.8:

Example 5.8 *We assume that axiom (#1), $RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$, has been modified by two sources at different times:*

ID	*trust*	*date*
#1	∞	*2009-09-10*
#1	*Nev*	*2009-09-09*

Then the provenance assignment for axiom $\#_1$ *is*

$$\{(trust, \infty), (date, \textit{2009-09-10})\} \oplus \{(trust, Nev), (date, \textit{2009-09-09})\} = \{(trust, \infty \oplus Nev), (date, \textit{2009-09-10} \oplus \textit{2009-09-10})\} = \{(trust, \infty), (date, \textit{2009-09-10})\}.$$

In contrast, if first Neverest and then DPA assert the same axiom, we would have

$$\{(trust, \infty), (date, \textit{2009-09-09})\} \oplus \{(trust, Nev), (date, \textit{2009-09-10})\}$$

As $\infty >_{trust} Nev$*, but* $\textit{2009-09-09} <_{date} \textit{2009-09-10}$*, these labels are incomparable and can not be further simplified.*

In order to accommodate such potentially conflicting provenance assignments about ontology axioms, we extend the semantics of provenance, which we have introduced in Section 5. For this purpose, we redefine the *prov* function of Definition 5.4, such that it uses $\oplus$ to merge provenance assignments in a preprocessing step. Afterwards, we have a unique provenance assignment again and apply Definition 5.4 as before.

Definition 5.7 (Provenance Extended) *Let allprov: axioms* $\rightarrow 2^{B_D}$ *be a function mapping from an axiom to all provenance assignments to that axiom in a provenance dimension* D*.*

Let prov *be a function mapping from an axiom to a provenance assignment in dimension* D*. The provenance of an axiom* A *wrt.* O *in* D *is obtained by computing a pinpointing formula* ϕ *of* A *wrt.* O *and obtaining* ψ *by replacing each axiom* A *in* ϕ *with* $\bigoplus(allprov(A))$ *and the logical operators* $\dot\vee$ *and* $\dot\wedge$ *with their corresponding operators in* D*. Then* prov*(A) is computed by evaluating* ψ*.*

Example 5.9 *Consider the provenance assignment of axiom #1 with conflicting provenance presented below and the provenance formula:*

$$ProvAss(\#1) \dot\wedge ProvAss(\#2)$$

the derived provenance is:

$$\{(trust, \infty), (date, \textit{2009-09-10})\} \oplus \{(trust, Nev), (date, \textit{2009-09-09})\} \dot\wedge \{(trust, \infty), (date, \textit{2009-09-09})\} = \{(trust, \infty \oplus Nev) \dot\wedge (trust, \infty), (date, \textit{2009-09-10} \oplus \textit{2009-09-10}) \dot\wedge (date, \textit{2009-09-09})\} = \{(trust, \infty \dot\wedge \infty), (date, \textit{2009-09-10} \dot\wedge \textit{2009-09-10})\}. = \{(trust, \infty), (date, \textit{2009-09-10})\}.$$

This definition of *prov* not only allows to aggregate provenance from multiple sources, but also to gracefully handle unknown provenance, i.e. situations where a knowledge source does not provide a label for some provenance dimension, in which case $\bot_D$ is assumed as a default, as introduced in Definition 5.2

Example 5.10 shows how our approach can be applied to our use case scenario.

Example 5.10 *Back to our scenario, DPA needed to verify that Bluewater indeed is a city in CA. Hence, they need to answer the query "bluewater: RealCity?" and to obtain provenance for the answer. The query results in the following provenance formula:*

$$ProvAss(\#3) \dot\wedge ProvAss(\#4) \dot\wedge ProvAss(\#6)$$

The resulting provenance label is:

$$\{(trust, \infty), (date, \textit{2009-09-10})\} \dot\wedge \{(trust, Nev), (date, \textit{2009-09-09})\} \dot\wedge (trust, NevWP), (date, \textit{2009-09-09})\} = \{(trust, inf_{Nev,NevWP}), (date, \textit{2009-09-10})\}.$$

Note that in Example 5.10 the labels of the modification date dimension are comparable, while the labels of the trust dimension are not. Hence, the resulting provenance label is a tuple of the infimum of Nev and NevWP in the trust dimension and the maximum of the dates in the assertion date component.

6. Computing Provenance Using Pinpoints

We now use our definitions of provenance in order to track down what recent addition to the knowledge base led to a desired or undesired effect. In the

Listing 1. ProvNaive(O, A, D)

```
pinpoints := GetAllPinpoints(A, O);
lab := {l |$\exists P \in$pinpoints: l =
      $\max_D$(prov(P))};
lab := {$\dot{\bigvee}_{l \in \mathrm{lab}} l$};
M := $\bigcup_{p \in \mathrm{pinpoints}}$p;
ProvNaive := (M, lab);
```

case of our collaborative ontology editing scenario, we may want to identify who was the least authoritative source that contributed most recently to an inconsistency.

The algorithms discussed in this section fulfill this purpose, i.e. given a query (an axiom), an ontology and a provenance dimension, they extract a subontology as explanation and compute the provenance label of the query.

As we can see above, Definition 5.4 relies on a pinpointing formula for the computation of provenance. Hence, we need to find all pinpoints to compute a pinpointing formula. Then we can immediately derive the provenance label.

6.1. *Naïve Algorithm*

A naïve evaluation of provenance for an axiom might compute all pinpoints and then evaluate the pinpointing formula. This strategy is illustrated in Algorithm `ProvNaive`. `ProvNaive` takes as parameters an ontology O, an axiom A and a provenance dimension D. It returns the provenance label of A wrt. O and a subontology containing all pinpoints for $O \models A$.

However, finding all pinpoints is a very expensive operation and this approach is not appropriate to be used in real use cases. Finding a pinpoint using a black-box approach may need exponentially many consistency checks in the underlying DL. Moreover, there can be exponentially many pinpoints.

6.2. *An Optimized Algorithm for Debugging with Provenance*

We present an optimized algorithm for deriving provenance, which performs significantly better in the average case than naïve implementations. It does not need to compute all pinpoints, and in fact not even precisely a pinpoint. Instead, we compute an approximation which is sufficient for deriving provenance.

The optimization is based on the assumption that the provenance dimension is a lattice $(B_D, \dot{\vee}_D, \dot{\wedge}_D)$ such that $a \dot{\vee}_D b = \sup(a,b)$ and $a \dot{\wedge}_D b = \inf(a,b)$ for $a, b \in B_D$ (or vice versa). Note that in the general case the interpretation of $\dot{\vee}_D$ and $\dot{\wedge}_D$ is independent as shown in Definition 5.1, i.e. they do not need to be dual. However, for optimization issues we assume that $\dot{\vee}_D$ and $\dot{\wedge}_D$ are no longer independent. This assumption is true for all provenance dimensions discussed above such as modification date, degree of trustworthiness and for all total orders. In this case, the pinpointing formula has the structure of a supremum of infima when expressed in disjunctive normal form.

Considering this assumption, we make the evaluation more efficient since we can exploit monotonicity properties of our provenance dimensions where applicable.

For instance, for a provenance dimension with a total order, once we find the pinpoint with the highest provenance label, we have also found the overall maximal provenance label. Thus, in many cases, we do not need to compute all pinpoints and we may restrict the computation of the pinpointing formula to those parts of an ontology that are relevant for the provenance computation given its particular lattice structure.

If the provenance dimension is a partial order, several pinpoints with incomparable provenance labels may be found. Thus, we need to find all pinpoints with maximal (minimal) provenance labels and merge the corresponding provenance labels to determine the resulting provenance label.

Without loss of generality, we only consider for our optimized algorithm the case that the corresponding pinpointing formulas have the structure of a supremum of infima. In this case the provenance label of a single pinpoint is the infimum of the labels of the axioms it contains, and the overall provenance label is the supremum of the labels of all such pinpoints. As a result, we do not need to take into account any axiom, which has a provenance label less than the infimum of the labels of the greatest pinpoints. For the case of a infimum of suprema, we simply need to invert all comparisons and replace min by max in Listings 2 and 3.

Likewise, we only consider consistent ontologies. The proposed algorithms handle inconsistent ontologies equally well, if the entailment check in line 10 of Listing 2 is negated.

The algorithm starts with a query of the form "$O \models A$?", as we focus on entailment checking here.

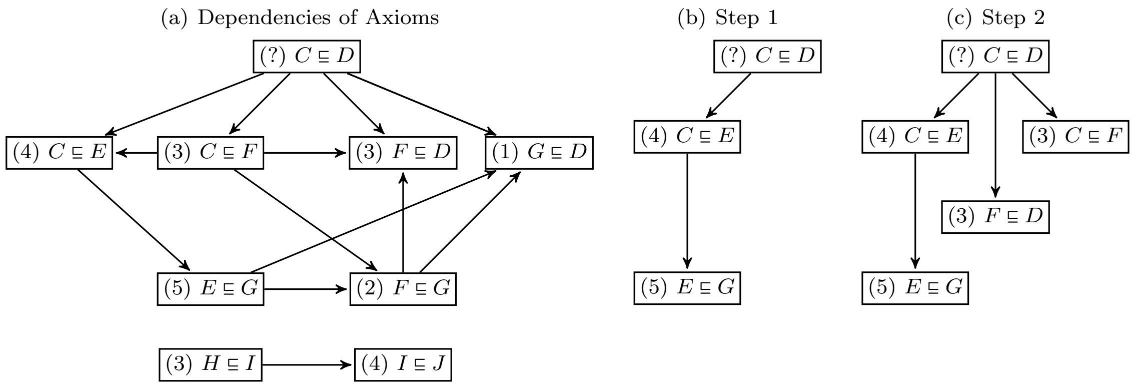

Fig. 3. Selection of Axioms by Optimized Algorithm.

In order to find those axioms that are relevant for the provenance computation given the query, the algorithm iteratively grows a subontology around A based on the *syntactic relevance* selection function. Intuitively, an axiom is syntactically relevant for another axiom, if it contributes to the definition of one of the concepts or properties in the other axiom. In Figure 3, a small example ontology is shown. The arrows represent syntactic relevance relationships between axioms. Assume the query is $C \sqsubseteq D$?. Then the two axioms at the bottom can not be relevant to the answer, because they are neither directly nor indirectly relevant to the query. The rest of the ontology contains three justifications for $C \sqsubseteq D$, namely $\{C \sqsubseteq E, E \sqsubseteq G, G \sqsubseteq D\}$, $\{C \sqsubseteq F, F \sqsubseteq G, G \sqsubseteq D\}$ and $\{C \sqsubseteq F, F \sqsubseteq D\}$

Definition 6.1 (Syntactic Relevance [20]) *An axiom B is directly syntactically relevant for an axiom A, if their clusters overlap, i.e. if they share a concept, role or individual. B is syntactically relevant for A, if it is directly syntactically relevant for A, or if B is syntactically relevant for C, which is syntactically relevant for A.*

We define a convenience function σ: Given A and an ontology O, $\sigma(A, O) = \{B \in O | B$ is directly syntactically relevant for $A\}$. The definition carries over to sets of axioms M: $\sigma(M, O) = \{B \in O | \exists A \in M : B$ is directly syntactically relevant for $A\}$

Using the syntactic relevance selection function, the algorithm `Prov(O, A, D)` computes the provenance label of $O \models A$ in dimension D, if $O \models A$. We start by determining the set of syntactically relevant axioms for $A = C \sqsubseteq D$ in line 3 ($C \sqsubseteq E$, $C \sqsubseteq F$, $F \sqsubseteq D$ and $G \sqsubseteq D$). Then we add the ones with the greatest provenance label to the module in line 4 ($C \sqsubseteq E$ in step1; $C \sqsubseteq F$ in step 2). In the inner loop we recursively add all syntactically relevant axioms for the new module, which do not decrease the provenance degree of the overall solution ($E \sqsubseteq G$ in step 2). Note that this is just an optimization step to avoid unnecessary entailment checks. It can be omitted to compute a more precise approximation of the pinpoint. The tradeoff is a possibly higher number of iterations and entailment checks. In each iteration we add axioms until we have found a module, which contains a pinpoint in line 10 (this is the case after step 2 of the example). The provenance label for A is the smallest provenance degree of the axioms in the module (hence, 3). We work with a set for *labels* to account for the fact that in the presence of partial orders, minimum and maximum are not unique. Hence, the *min* function used in line 11 is the set version, which returns the set of smallest elements of a set. For total orders, *label* always is a singleton.

Note that our optimization strategy is strongly related to two issues: (a) the syntactic relationships among the pinpointing axioms, and (b) the size of pinpoints. The closer the pinpointing axioms are syntactically correlated and the smaller the pinpoints are, the faster the module can be built and the provenance label be derived. By growing only a small module of the ontology and avoiding to use the instantiated ontology as a whole for approximating pinpoints, we save lots of entailment checks which are very expensive.

Theorem 6.1 *Let O be an ontology, A an axiom such that $O \models A$ and D a provenance dimension, which is a total order. `Prov(O, A, D)` computes the provenance degree of $O \models A$ in dimension D.*

Proof: First we show that `Prov` terminates. The inner loop terminates when M-$A \models A$. At each iter-

Listing 2. Prov(O, A, D)

```
1  M := {$A$};
2  repeat
3    syn := $\sigma$(M, O);
4    add := {$B_1 \in$ syn |$\nexists B_2 \in$
          syn : prov($B_1$)$<_D$prov($B_2$)}
5    repeat
6      M := M $\cup$ add
7      syn := $\sigma$(M, O)
8      add := {$B_1 \in$ syn |$\exists B_2 \in$
          M: prov($B_1$)$\geq_D$prov($B_2$)}
9    until (add $\subseteq$ M)
10 until (M-A $\vDash A$);
11 lab := min$_D$({prov(B)|$B \in M$})
12 Prov := (M, lab);
```

ation *add* is assigned to a subset of O, which is syntactically relevant to *module* and has the greatest provenance degree. Consequently, *module* is a subset of O. As O is finite and $O \vDash A$, the loop must eventually terminate. In this case *M-A* indeed $\vDash A$ and *lab* = *min*$_D$({*prov(B)*|$B \in M$}).

It remains to show that if `Prov` terminates, it returns the correct provenance degree of $O \vDash A$. Remember that we are looking for the pinpoint with the highest provenance label and that the provenance label of the pinpoint is the infimum of the provenance labels of its axioms. M only contains axioms which are syntactically relevant for A. When the outer loop terminates, M contains a (superset of a) greatest pinpoint of A wrt. O. Assume there is a pinpoint P, which has not been found and has a greatest provenance label. Then it must already be part of *module*: All axioms in a pinpoint of A must be syntactically relevant for A. Moreover, $\forall B \in P$: prov$(B) >_D$ *lab*. M contains all such axioms, which are directly syntactically relevant for A, or which are indirectly syntactically relevant via axioms with provenance labels $\geq_D$ *lab*. Hence, M must already contain P. Refutation.

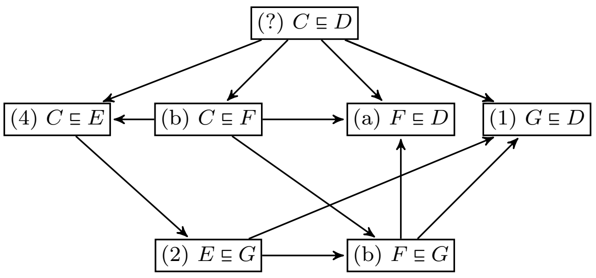

Fig. 4. Justifications with incomparable provenance.

For partial orders the resulting *label* may contain multiple elements which need to be merged. Figure 4 illustrates a case where we have a complex dimension composed of numbers and letters and their natural orders and multiple relevant pinpoints, which together result in a provenance label of $\inf_{1,a}$. `ProvPartial` computes accurate labels for partial orders. It uses `Prov` to compute an approximation first. Although there might be multiple pinpoints with incomparable provenance labels, `Prov` stops after the first pinpoint is contained in the approximation. Therefore we need to further extend the approximation, such that all possibly relevant axioms are included. If the label returned by `Prov` is a singleton, there is a unique maximal pinpoint (lines 2 and 3). Otherwise, we compute the minimum of all elements of *lab* (line 4), which is the lower bound D we need to consider. For example, as numbers and letters are incomparable, the lower bound for a and 2 is $\inf_{1,a}$. This means we could ignore any axiom, which does not have a provenance label (and hence defaults to $\bot$), but need to consider $G \sqsubseteq D$. Every pinpoint with a provenance label equal to or less than *min* is also less than the label of the pinpoint we have already found in `Prov`, and hence we can ignore it. We now extend our approximation with all axioms, which are syntactically relevant to A and, if they are indirectly relevant, are connected to A only through axioms, which have a provenance label greater than *min*. The loop in lines 5 to 9 works analogously to lines 5 to 9 in `Prov`. Finally, we compute the correct label using `ProvNaive`.

Listing 3. ProvPartial(O, A, D)

```
1  (M, lab) := Prov(O, A, D);
2  if (|lab| = 1) then
3    ProvPartial := (M, lab);
4  min := $\dot{\bigwedge}_{l \in lab}$ l;
5  repeat
6    N := M
7    syn := $\sigma$(M, O);
8    M := M $\cup$\{$B \in$ syn | prov(B)$>_D$min\};
9  until (M = N)
10 ProvPartial := ProvNaive(M, A, D);
```

Theorem 6.2 *Let O be an ontology, A an axiom such that $O \vDash A$ and D a provenance dimension, which is a partial order. `ProvPartial(O, A, D)` computes the provenance degree of $O \vDash A$ in dimension D.*

In line 9 we use the naïve algorithm. Therefore we only need to show two things: (a) If `Prov` returns a singleton, it is indeed correct also for partial orders and (b) lines 4 to 9 indeed compute a module, which contains all axioms relevant for computing the correct solution using the naïve algorithm.

(a) In line 3 and 4 of `Prov` exactly those axioms are added to M, which are minimal in the current syntactic relevance set `syn`. Hence, if incomparable provenance degrees occur, all of them are selected. They are also preserved in line 11, as there is no unique minimum in this case. Thus, if a unique minimum is returned, then it must in fact be equal or greater than the provenance labels of all other pinpoints.

(b) Assume there is an axiom B, which is contained in some pinpoint P, which is relevant for `lab` and missing in M. We consider two cases:

(b1) $\dot{\bigwedge}_{p \in P} \text{prov}(p) \leq \texttt{min}$. That means some prov($p$) is less or equal `min`. Then P is not relevant. Refutation.

(b2) Hence, *prov(p)* must be greater than `min` for all $p \in P$. Then all p and hence also B have been selected in lines 5 to 9. Refutation.

Figure 5 presents a diagram showing the relations between the set of axioms of the ontology (set O), the set of axioms of the relevant pinpoints (set P), and the set of axioms of the module retrieved by our algorithm (set R). Note that the algorithm retrieves some of the pinpoint axioms. Axioms that are part of the pinpointing formula and are retrieved by the algorithm (the overlap of the sets P and R) correspond to those that are relevant and necessary for deriving provenance.

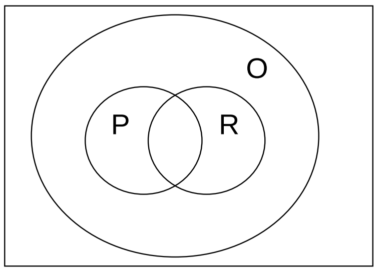

Fig. 5. Relations between the set of axioms of the ontology (O), the axioms in all pinpoints (P) and the axioms retrieved by the algorithm (R).

7. Complexity

The worst case complexity of both the naïve and the optimized approach for computing provenance is equivalent to the computation of all pinpoints in the underlying logic as the pinpointing formula can be evaluated in polynomial time. If it is expressed in normal form, however, the size of the formula can blow up exponentially. Approaches for computing pinpoints like [7] which, rather than representing pinpoints formula in a normal form, derive a compact representation of the pinpoints formula benefit the computation of provenance since they avoid exponential blowup.

While in the worst case the complexity of computing provenance is the same as the complexity of finding all pinpoints and hence quite high, in the average case we can do much better using the optimized algorithm as we show in Section 8.

8. Evaluation

The prototype of the optimized provenance computation algorithm (`Prov`) has been implemented in Java 1.6 using Pellet 2.0.0 [4] and OWL API trunk revision 1310 [5]. `Prov` is available as an open source implementation at `http://west.uni-koblenz.de/Research/MetaKnowledge`.

We have performed our experiments using two groups of ontologies. The first experiment is composed of standard ontologies for debugging ontologies that have already been used for testing the computing time of laconic justifications in [18]. These ontologies have been used in [18] to demonstrate the efficiency of traditional computation of pinpoints. The main goal of this experiment is to show that our optimized algorithm speeds up the performance of provenance computations compared to the standard computation of pinpoints.

The second experiment uses real-world living ontologies from Bioportals containing real change logs with timestamps and editors. The main goal of this experiment is to test the applicability and scalability of our approach for computing provenance in real time. For both experiments we use a virtual machine with 2GB RAM and a single Intel(R) Xeon(TM) CPU core with 3.60GHz.

Experiment I - Debugging Ontologies

In this experiment, we compare our optimized algorithm to the naïve approach (baseline) based on full axiom pinpointing. For this experiment, we have used the ontologies from Table 4. This dataset has

[4] Pellet: OWL 2 Reasoner for Java (http://clarkparsia.com/pellet)
[5] The OWL API (http://owlapi.sourceforge.net/)

been already used in [18] to demonstrate the efficiency of traditional computation of pinpoints, and it offers a range of varying complexities and pinpoint sizes.

Since for these ontologies no provenance information is available, we have generated artificial provenance in two ways:

random - ontologies 1-6 We have augmented the original ontologies by randomly assigning timestamp values and degrees of trustworthiness to the axioms.

cluster - ontologies 7-12 We have augmented the original ontologies by assigning similar timestamp values and degrees of trustworthiness to such clusters of axioms which are syntactically relevant to each other. This reflects the fact that a user usually does not do random modifications, but changes a part of the ontology focused around a certain class or property.

As the ontologies 1 to 12 are inconsistent, our implementation focuses on pinpoints for inconsistencies instead of subsumptions. For this reason, the termination condition in line 8 of `Prov` has been changed to a consistency check, i.e. $M - A \models \neg A$.

For each ontology, we have measured the time needed to compute the provenance for an inconsistency in average over all inconsistent classes of an ontology. In order to investigate the influence of composing dimensions, we have performed the experiment for each ontology with a single and with two provenance dimensions (see Section 5.2).

We use as baseline the naïve approach based on full axiom pinpointing the black box algorithm implementation in the OWL API for computing all pinpoints.

The results of the evaluation are presented in Figure 6 and Table 5. In Figure 6, we have normalized the results to the processing time of the naïve approach and used a logarithmic scale. The absolute numbers ranged over three orders of magnitude. Our optimized algorithm performs significantly better for all cases and scales very well. Note that in some ontologies we can find some very large pinpoints and some entailments with high numbers of pinpoints. In contrast, some other ontologies contain relatively small and few pinpoints.

As shown in Figure 6 ontology 5, which is an annotated version of the Chemical ontology, required the most computation time in the naïve approach as it contains some very large pinpoints and some entailments with high numbers of pinpoints. Specially for such cases, our approach has shown its potential.

As already expected, the optimized cluster ontology group (ontologies 7-12) has performed better than the optimized random ontology group (ontologies 1-6). The reason is that for the optimized cluster group we have assigned similar provenance labels to syntactically related axioms. Since in our approach, axioms that are syntactically relevant to the query are selected first, for the optimized cluster group the module could be built faster than for the random group. This increases the performance for computing provenance since axioms belonging to a common pinpoint are in general syntactically related to each other. As instance, ontology 10 has shown better performance as ontology 4 since the relevant axioms could be selected first, and thus a smaller module could be built.

Furthermore, we observe that debugging with many independent dimensions can have a negative performance impact due to lots of incomparable labels in the composed dimension, but in most cases still performs significantly better repeating the computation for two atomic dimensions.

Experiment II - Living Ontologies

In our second experiment, we evaluate if our approach is fast enough to support user assessments of the reliability of inferences in real time also for real world, large scale data. For this experiment, we have used the two real-world ontologies shown in Table 6.

The BIO ontology is a mapping ontology for the Open Biomedical Ontologies (BIO) repository of the US National Cancer Institutes Center for Bioinformatics [5], ontology 13. The BIO ontology comes with real change logs containing timestamps and editor information for each axiom, which we use to evaluate our approach.

The OBI ontology is the Ontology for Biomedical Investigations [6], ontology 14. This is an integrated ontology for the description of life-science and clinical investigations. It supports the consistent annotation of biomedical investigations, regardless of the particular field of study. The OBI ontology has been developed in collaboration with groups representing different biological and technological domains involved in Biomedical Investigations. In the OBI ontology, changes are tracked in change logs with timestamps and provenance information.

[6] OBI: The Ontology for Biomedical Investigations (http://obi-ontology.org/)

	Ontology	Expressivity	Total Axioms	No.Unsat. Classes	Av. of Pinpoints per Unsat. Class
1,7	People	$\mathcal{ALCHOIN}$	372	1	1
2,8	MiniTambis	$\mathcal{ALCN}$	400	30	1
3,9	University	$\mathcal{SOIN}$	92	9	1
4,10	Economy	$\mathcal{ALCH(S)}$	2.330	51	1
5,11	Chemical	$\mathcal{ALCHF}$	192	37	11
6,12	Transport	$\mathcal{ALCH}$	2.178	62	2

Table 4
Ontologies used in the experiments.

	Ontology	Baseline	Av. Size of Pinpoints per Unsat. Class	Provenance. *Date* Dimension	Av.Size of Module per Unsat. Class	Provenance. *Date* and *Trust* Dimensions	Av.Size of Module per Unsat. Class
1	People Random	312	4	**141**	15	141	15
2	MiniTambis Random	141	7	**18**	22	22	22
3	University Random	53	4	**8**	18	8	18
4	Economy Random	395	3	**23**	57	27	58
5	Chemical Random	3239	7	**32**	126	38	126
6	Transport Random	1165	5	**23**	70	29	71
7	People Cluster	141	4	**16**	15	16	15
8	MiniTambis Cluster	127	7	**11**	22	11	22
9	University Cluster	52	4	**7**	18	7	18
10	Economy Cluster	386	3	**4**	19	5	19
11	Chemical Cluster	3236	7	**32**	126	37	126
12	Transport Cluster	1121	5	**28**	99	34	102

Table 5
Average time to compute provenance for an inconsistency of an ontology (ms).

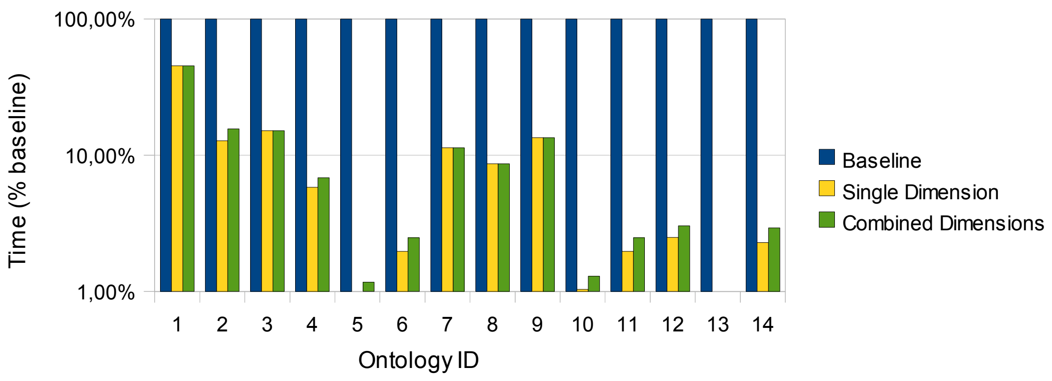

Fig. 6. Average time to compute provenance for an inconsistency of an ontology (in logarithmic scale). The computation of provenance for ontology 5, 10 and 13 scales orders of magnitude better than a naïve implementation algorithm and for that reason, they are not displayed on the graph.

For both ontologies, we mapped information about editors to degrees of trustworthiness as defined in Section 3.3, and extracted the timestamps for the modification dates.

We compared our approach to the naïve approach for computing all pinpoints. For each ontology, we have evaluated the time needed to compute the provenance for an inconsistency in average over all inconsistent classes of an ontology.

Since ontology 13 and 14 are consistent, we have generated random queries as follows: For randomly selected disjoint classes A and B we introduced a new class C such that $A \sqsubseteq C$ and $B \sqsubseteq C$ and used these new axioms as queries. For this reason, we could use exactly the same implementation as for Experiment I. We have performed the experiment for each ontology with a single and with two dimensions to investigate the influence of composing dimensions.

The results of the evaluation are presented in Figure 6 and Table 7. For ontology 13 and 14 our algorithm is 97-99% faster than the baseline. For the BIO ontology, which contains 660.915 axioms, provenance can be computed in under 8 seconds with our algorithm, which is fast enough for interactive applications. In contrast, the baseline approach takes almost 22 minutes.

The missing bars for ontology 13 in Figure 6 are due to the extreme differences in runtime. For the relatively small ontologies used for experiments 1 to 12, existing pinpointing algorithms perform pretty well. However, for the extremely large ontologies 13 and 14, our optimizations show their full potential.

Using our approach, the time needed to compute provenance mainly scales with the size of pinpoints, rather than with the size of the ontology. Compared to the size especially of ontology 13, our machine had a rather small main memory (most of it was already necessary to just classify the ontology using Pellet). We expect even better performance, when memory management is less an issue.

Debugging with many independent dimensions can have a negative performance impact due to lots of incomparable labels in the composed dimension. Results for the real world data from the BIO and the OBI ontologies show that in reality this is less a problem, as modification dates and editors are not independent there. In fact, the lack of independence of the provenance dimensions is an advantage in practice. The results for ontology 13 are even better for the complex dimension than for the atomic dimensions. This shows that our assumption for the clustered random data obviously is correct.

Precision and Recall

Finally, we have analyzed the precision and recall of our approximation. Precision is defined here as the number of retrieved axioms that are relevant for deriving provenance divided by the total number of retrieved axioms. Recall is defined as the number of retrieved axioms that are relevant for deriving provenance divided by the total number of available axioms of the ontology that are relevant for deriving provenance.

The recall clearly always is 1.0, as assured by the inner loop in lines 5 to 9 in `Prov` and `ProvPartial`. We retrieve all axioms of the pinpoint(s) that are relevant and necessary for computing the provenance label (see Figure 5). However, we have to tolerate a certain percentage of false positives (low precision), since not all retrieved axioms are relevant for deriving provenance.

Table 8 shows that for ontologies 1-3 and 7-9 our approach leads to high precision. However, for the ontologies 4-6, 10-12, as well as for both real-world ontologies 13 and 14, our approach achieves low precision. The reason for this discrepancy is that the latter ontologies are highly axiomatized ontologies with many syntactic dependencies among axioms. Thus, using the syntactic relevance function our algorithm also retrieves many irrelevant ones. As we have mentioned before, we can trade the runtime optimization in the inner loop of the algorithm for higher precision of the module. In other words, we have a trade-off between (low) precision and (good) performance.

Table 8 shows the relation between the size of the module retrieved by the algorithm `Prov` presented in Section 6.2 and the total number of relevant axioms for deriving provenance. We measure the average module size for computing the provenance and the average number of axioms in all relevant pinpoints for an inconsistency. The numbers are averaged over all inconsistent classes of an ontology.

9. Related Work

Initial work towards the approach presented in this paper has been published in [31]. We extend the work in [31] with an optimized algorithm for computing provenance and a comprehensive evaluation. Moreover, the composition of multiple provenance

	Ontology	Expressivity	Total Axioms	No. Unsat. Classes	Av. of Pinpoints pro Unsat. Class
13	BIO	$\mathcal{SHOIN}$(D)	660.915	1	1
14	OBI	$\mathcal{SHOIN}$(D)	16.415	1	1

Table 6
Ontologies used in the experiments.

	Ontology	Baseline	Av. Size of Pinpoints pro Unsat. Class	Provenance. *Date* Dimension	Av.Size of Module pro Unsat. Class	Provenance. *Date* and *Trust* Dimensions	Av.Size of Module pro Unsat. Class
13	BIO	1.273.281	3	7.514	33	**7.306**	33
14	OBI	24.474	3	**660**	150	717	150

Table 7
Absolute times needed for the experiments (ms).

	Ontology	Average Axioms	Average Module Size	Precision
1	People Random	**4**	**15**	**0.3**
2	MiniTambis Random	**7**	**22**	**0.3**
3	University Random	4	18	0.2
4	Economy Random	3	57	0.05
5	Chemical Random	7	126	0.06
6	Transport Random	5	70	0.07
7	People Cluster	**4**	**15**	**0.3**
8	MiniTambis Cluster	**7**	**22**	**0.3**
9	University Cluster	4	18	0.2
10	Economy Cluster	3	19	0.16
11	Chemical Cluster	7	126	0.06
12	Transport Cluster	5	102	0.05
13	BIO	3	33	0.09
14	OBI	3	150	0.02

Table 8
Average of the relevant axioms for deriving provenance vs the average of the module size.

dimensions and comparisons to other approaches has been added and the formalization has been extended correspondingly.

[26] propose infrastructure for trust on the semantic web such as portable explanations and service registries. While [26] and our approach can augment each other, ours is significantly different, because we provide a flexible framework for arbitrary provenance dimensions. Furthermore, we do not need to generate accurate proofs to track provenance.

In [6] the authors propose an approach for computing boundaries for reasoning in sub-ontologies to enforce access restrictions. Provenance is used as criterion for pre-computing sub-ontologies during the development phase where the consequences still follow for pre-determined axioms. Two optimizations for reasoning are proposed: An extension of the hitting set tree (HST) algorithm for general lattices, which is close to state of the art pinpointing algorithms and binary search for total orders. For large ontologies, [6] uses a modularization step, before applying the actual reasoning. The evaluations of [6] and our approach are not directly comparable, as [6] focuses on either large ontologies with low expressivity ($\mathcal{EL}+$) or small ontologies with high expressivity. Moreover, the modularization step is not included in the evaluation. [22] extends the approach towards explanations and debugging of access rights. In contrast, our approach is built on modularization at its core without a need for preprocessing and generates explanations while computing provenance. We have shown that our approach scales very well even for large and very expressive ($\mathcal{SHOIN}(D)$) ontologies. Besides a single access rights dimension, we discuss the management of the various provenance dimensions relevant for the tracking of ontology dynamics. The two approaches might be joined in future work by using the HST algorithm from [6] to replace the naïve algorithm used in `ProvPartial`. Such a combined algorithm would have desirable features of both approaches.

[12] introduces a trust framework based on Bayesian Description Logics that allows to compute a degree of inconsistency over a probabilistic knowledge base. They consider pinpoints as possible worlds for an axiom and derive for each possible word a probability measure. The degree of incon-

sistency of a knowledge base is then computed as the sum of the probabilities associated with possible worlds that are inconsistent. Due to scalability reasons, the proposed trust computation model operates on a random sample of justifications. Our optimized algorithm does not deal with the sum of all justifications, and thus we do not need to compute all justifications to determine the degree of inconsistency for a query. We are interested in deriving the provenance labels of the changes which led to the inconsistency.

In [32], the authors propose a framework which allows to select the most probable diagnosis (pinpoint) to repair an inconsistency. Basically, it creates a sequence of queries which is used to reduce the set of diagnosis until it identifies the target one. The target diagnosis is the one which contains all entailments of the target ontology. To select the best queries to be posted next, the algorithm predicts the information gain of a query result, i.e. the result which minimizes the entropy. Our approach has a different goal when using provenance information for reasoning. Since we do not aim to compute the most probable possible world over all possible worlds but we want to select some specific axioms based on their provenance, we do not need to compute all pinpoints to track provenance.

Further related work can be grouped into the following categories: (i) Extensions of description logics with a particular provenance dimension, especially uncertainty. (ii) General provenance for query answering with algebraic query languages. (iii) Extensions of description logics with general provenance and (iv) provenance for other logical formalisms.

ad (i) Several multi-valued extensions of description logic have been proposed: [24] propose fuzzy and probabilistic extensions of the DLs underlying the web ontology language OWL. [28] describe an extension towards a possibilistic logic. Another extension towards multi-valued logic is presented by [30]. They aim at trust and paraconsistency instead of uncertainty. OWL-2 is extended to reasoning over logical bilattices. Bilattices, which reflect the desired trust orders, are then used for reasoning. The approach allows for paraconsistent reasoning with OWL taking into account trust levels. [25] provide an extension to reasoning in OWL with paraconsistency by reducing it to classical DL reasoning. On the level of RDF, [11] "color" triples to track information sources. All of these approaches have in common that they modify the character of models in the underlying description logic, e.g. to fuzzy or possibilistic models. In our approach in contrast, we reason on a meta level: While the underlying model remains unchanged, we compute consequences of annotations on axioms. This meta level reasoning is not possible in the approaches proposed above. Unlike general provenance, these approaches are tailored to a specific need and most of them do not respect the provenance dimensions needed for tracking ontology dynamics. In [4], the authors propose a temporal extension of description logics which is a combination of the epistemic modal logic $S5$ with the standard DL $\mathcal{ALCQI}$. $S5_{\mathcal{ALCQI}}$ that can represent rigid concepts and roles and allows one to state that concept and role memberships change in time(but without discriminating between changes in the past and future). This approach weakens the temporal dimension to the much simpler S5, but can nevertheless show that adding change (i.e., timestamping on entities, relationships and attributes) pushes the complexity of $\mathcal{ALCQI}$ from ExpTime-complete to 2-ExpTime-hard. Likewise in [3] the authors propose a temporal extension of description logic $T_{\diamond}DL - Lite_{bool}$ that can additionally capture some form of evolution constraints. Both works aim at analysing the satisfiability problem (decidable or undecidable) for temporal extensions of DL. Since we do not work with any temporal extension of DL, our problem is restrict to the underlying logic.

ad (ii) Provenance for algebraic base languages has been proposed by various authors, for example for the Semantic Web Query Language SPARQL [10,16,23] and for relational databases [9]. While the actual provenance formalisms are comparable to ours, the underlying languages are of lower expressivity, typically Datalog. Provenance in these languages can directly be evaluated using the tree shaped algebraic representations of a query, which is not possible in description logics.

ad (iii) [33] propose a provenance extension of OWL, which is also based on annotation properties. Even though provenance can be expressed in ways comparable to ours, it has a rather ad-hoc semantics, which may differ from query to query. In our approach, provenance and classical reasoning take place in parallel. Hence, we can answer queries such as "Give me all results with a confidence degree of $\geq x$". In contrast, reasoning on the ontology and meta level is separated in [33]. As a result, [33] allows for queries such as "Give me all results, which are based on axioms with a confidence degree of $\geq x$". Although this difference might seem quite subtle, depending on the provenance dimension these queries may have dramatically different results.

ad (iv) [8] propose an extension of Datalog with weights, which are based on c-semirings and can be redefined to reflect various notions of trust and uncertainty. Our provenance dimensions are similar to c-semirings, but additionally allow to handle conflicting provenance using a third operator. C-semirings have been investigated in great detail and have some desirable properties, such as the fact that the cartesian product of two c-semirings again is a c-semiring. Although based on a less strictly defined algebraic structure, our composition of provenance dimensions described in Section 5.2 follows a similar idea.

10. Conclusion

When querying and reasoning while integrating knowledge from different sources on the Semantic Web, applications need to track the provenance of axioms in living ontologies. Users need tool support for judging the usability and trustworthiness of ontological data, as well as engineering tools supporting the collaborative development of living ontologies. In dynamic ontologies ever changing criteria may be necessary to establish trust or locate bugs.

We have defined *provenance* as a very flexible and extensible framework for expressing information about evolving ontologies. At the same time, provenance has a clear semantics for reasoning and debugging with such dynamic ontologies. We have shown that this additional information does not only provide value to the end user. Further, it can be used to significantly speed up debugging processes by rapidly approximating a solution.

In this paper we present our framework restricted to ontology diagnosis scenarios. Our work, however, introduces an approach for provenance querying under a variety of scenarios (that consider many of the dimensions of provenance) such as to restrictions of access rights, knowledge validity when the truth of knowledge changes with time, and inferring trust value.

In future work, we will apply our approach to provenance to other logical formalisms beyond DL $\mathcal{SRIQ}(\mathcal{D})$.

References

[1] A. Kalyanpur et al. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL. Technical report, UMIACS 2006, 2006.

[2] B. Thomas Adler and L. de Alfaro. A content-driven reputation system for the wikipedia. In *WWW*, pages 261–270. ACM, 2007.

[3] A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Dl-lite with temporalised concepts, rigid axioms and roles. In *FroCoS*, pages 133–148. Springer, 2009.

[4] A. Artale, C. Lutz, and D. Toman. A description logic of change. In *IJCAI*, pages 218–223. Morgan Kaufmann Publishers Inc., 2007.

[5] B. Smith et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[6] F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In *ISWC*, LNCS, pages 49–64. Springer, 2009.

[7] F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010.

[8] S. Bistarelli, F. Martinelli, and F. Santini. A Semantic Foundation for Trust Management Languages with Weights: An Application to the RT Family. In *ATC*, pages 481–495. Springer, 2008.

[9] P. Buneman, S. Khanna, and W. Chiew Tan. Why and where: A characterization of data provenance. In *ICDT*, LNCS, pages 316–330. Springer, 2001.

[10] R. Dividino, S. Sizov, S. Staab, and B. Schueler. Querying for provenance, trust, uncertainty and other meta knowledge in rdf. *Journal of Web Semantics*, 7(3):204–219, 2009.

[11] G. Flouris, I. Fundulaki, P. Pediaditis, Y. Theoharis, and V. Christophides. Coloring rdf triples to capture provenance. In *ISWC*, LNCS, pages 196–212. Springer, 2009.

[12] A. Fokoue, M. Srivatsa, and R. Young. Assessing trust in uncertain information. In *ISWC*, LNCS, pages 209–224. Springer, 2010.

[13] J. Golbeck. Inferring reputation on the semantic web. In *WWW*. ACM, 2004.

[14] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance Semirings. In *PODS*, pages 31–40. ACM, 2007.

[15] H. Halpin. Provenance: The missing component of the semantic web. In *SPOT*. CEUR, 2009.

[16] O. Hartig. Querying trust in rdf data with tsparql. In *ESWC*, LNCS, pages 5–20. Springer, 2009.

[17] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[18] M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. In *ISWC*, LNCS, pages 323–338. Springer, 2008.

[19] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible sroiq. In Patrick Doherty, John Mylopoulos,

and Christopher A. Welty, editors, *KR*, pages 57–67. AAAI Press, 2006.

[20] Q. Ji, G. Qi, and P. Haase. A relevance-directed algorithm for finding justifications of dl entailments. In *ASWC*, LNCS, pages 306–320. Springer, 2009.

[21] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of owl dl entailments. In *ISWC/ASWC*, LNCS, pages 267–280. Springer, 2007.

[22] M. Knechtel and R. Peñaloza. A generic approach for correcting access restrictions to a consequence. In *ESWC*, LNCS, pages 167–182. Springer, 2010.

[23] N. Lopes, A. Polleres, U. Straccia, and A. Zimmermann. Anql: Sparqling up annotated rdfs. In *ISWC*, LNCS, pages 518–533. Springer, 2010.

[24] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the Semantic Web. *Journal of Web Semantics*, 6(4):291–308, 2008.

[25] Y. Ma, P. Hitzler, and Z. Lin. Algorithms for Paraconsistent Reasoning with OWL. In *ESWC*, LNCS, pages 399–413. Springer, 2007.

[26] D. L. McGuinness and P. Pinheiro da Silva. Infrastructure for web explanations. In *ISWC*, pages 113–129. IEEE Computer Society, 2003.

[27] B. Motik. Representing and querying validity time in rdf and owl: a logic-based approach. In *ISWC*, LNCS, pages 550–565. Springer, 2010.

[28] G. Qi, J. Z. Pan, and Q. Ji. Extending Description Logics with Uncertainty Reasoning in Possibilistic Logic. In *ECSQARU*, pages 828–839. Springer, 2007.

[29] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[30] S. Schenk. On the Semantics of Trust and Caching in the Semantic Web. In *ISWC*, volume 5313 of *LNCS*, pages 533–549. Springer, 2008.

[31] S. Schenk, R. Dividino, and S. Staab. Reasoning with provenance, trust and all that other meta knowlege in owl2. In *SWPM*. CEUR, 2009.

[32] K. Shchekotykhin and G. Friedrich. Query strategy for sequential ontology debugging. In *ISWC*, LNCS, pages 696–712. Springer, 2010.

[33] D. T. Tran, P. Haase, B. Motik, B. Cuenca-Grau, and I. Horrocks. Metalevel Information in Ontology-Based Applications. In *AAAI*, pages 1237–1242. AAAI Press, 2008.