Fast Computation of Explanations for Inconsistency in Large-Scale Knowledge Graphs

Trung-Kien Tran
Bosch Center for Artificial Intelligence
Renningen, Germany
trungkien.tran@de.bosch.com

Mohamed H. Gad-Elrab
Bosch Center for Artificial Intelligence
Renningen, Germany
Max-Planck Institute for Informatics
Saarbrücken, Germany
gadelrab@mpi-inf.mpg.de

Daria Stepanova
Bosch Center for Artificial Intelligence
Renningen, Germany
daria.stepanova@de.bosch.com

Evgeny Kharlamov
Bosch Center for Artificial Intelligence
Renningen, Germany
evgeny.kharlamov@de.bosch.com

Jannik Strötgen
Bosch Center for Artificial Intelligence
Renningen, Germany
jannik.stroetgen@de.bosch.com

ABSTRACT

Knowledge graphs (KGs) are essential resources for many applications including Web search and question answering. As KGs are often automatically constructed, they may contain incorrect facts. Detecting them is a crucial, yet extremely expensive task. Prominent solutions detect and explain inconsistency in KGs with respect to accompanying ontologies that describe the KG domain of interest. Compared to machine learning methods they are more reliable and human-interpretable but scale poorly on large KGs. In this paper, we present a novel approach to dramatically speed up the process of detecting and explaining inconsistency in large KGs by exploiting KG abstractions that capture prominent data patterns. Though much smaller, KG abstractions preserve inconsistency and their explanations. Our experiments with large KGs (e.g., DBpedia and Yago) demonstrate the feasibility of our approach and show that it significantly outperforms the popular baseline.

ACM Reference Format:
Trung-Kien Tran, Mohamed H. Gad-Elrab, Daria Stepanova, Evgeny Kharlamov, and Jannik Strötgen. 2020. Fast Computation of Explanations for Inconsistency in Large-Scale Knowledge Graphs. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3366423.3380014

1 INTRODUCTION

Motivation. A Knowledge Graph (KG) describes facts about entities by labeling entities with classes and interconnecting them via binary relations. For instance, the top of Figure 1 depicts a KG with facts about companies and locations, e.g., `Nokia` is a `PopularName`, and it is a customer of `IBM`. Prominent examples of large-scale KGs include DBpedia [19], NELL [23], Yago [31], Freebase [3], and Wikidata [34] – all contain millions of facts. KGs power a wide variety of important applications including Web search [5] and question answering [6]. While data quality of a KG is crucial for such applications, unfortunately large-scale KGs are often inaccurate. Indeed, they are constructed using error-prone methods such as information extraction (e.g., Yago), crowd-sourcing (e.g., Wikidata) or curated using embedding-based approaches (see [36] for an overview). These methods inevitably introduce erroneous information, e.g., incorrectly disambiguated entities or faulty relations.

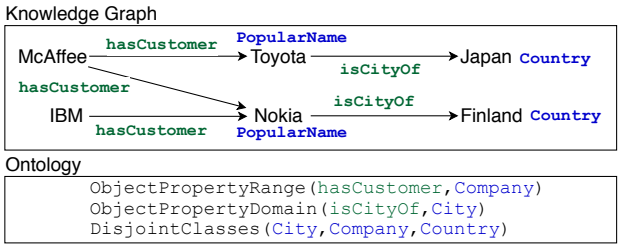

Figure 1: A sample KG and ontology.

A common task in such settings is that of *detecting incorrect facts* in KGs. Although many works on this topic have focused on machine-learning techniques (e.g., [15, 27]), methods that rely on symbolic reasoning (e.g., [25, 32]) have shown the benefit for this task due to their accuracy and human-interpretability [26]. One prominent such method for detecting incorrect facts in KGs is to compute inconsistency explanations of the KG with respect to a manually crafted ontology [14], which is a set of axioms that capture the relevant domain of interest and constraints enforcing that some statements must not occur together. An example ontology presented at the bottom of Figure 1 states that the range of `hasCustomer` relation is a `Company`, the domain of `isCityOf` relation is a `City`, and the classes `City`, `Company` and `Country` are disjoint. Such ontologies are usually carefully created by domain experts, and the errors in the KGs are typically characterized by whether there exist triples in the KG that contradict with the accompanied ontology. It is easy to verify that our sample KG contradicts the ontology, as `Toyota` and `Nokia` belong to disjoint classes `Company` and `City`. A minimal set of axioms causing this is an *inconsistency explanation*.

Detecting and explaining inconsistency in KGs enhanced with ontologies are fundamental and well-studied knowledge management tasks [13, 26]. It is well-known that finding all and even some explanations is computationally demanding and thus does not scale

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380014

for large real-world KGs [28]. This holds even for light-weight ontology languages that admit scalable processing since the number of explanations can be huge: in DBpedia there are at least 3,500,000 such explanations [28].

General Idea. We propose a novel method to compute inconsistency explanations for large-scale KGs. The key idea of our method is to identify an ontology language and construct a compressed representation, called *abstraction* (*abstract KG*), of the original KG such that for any ontology written in that language, the following requirements are satisfied:

- **[R1]** *preserve KG's consistency*: the original KG is consistent with respect to the ontology iff its abstraction is so,
- **[R2]** *preserve KG's explanations*: inconsistency explanations for the abstraction can be used to obtain exactly all inconsistency explanations for the original KG, and
- **[R3]** *be smaller than KG*: the size of the abstraction is much smaller than the size of the original KG.

Then, instead of checking consistency and computing the explanations directly on the original KG (w.r.t. an ontology), we perform such operations on its abstraction. The requirements R1–R2 guarantee the correctness of our method while R3 ensures its scalability.

Contributions. We show the existence of such abstractions for a fairly expressive fragment of the Web Ontology Language (OWL 2). More precisely, our contributions are as follows:

- Following the requirements R1–R3 we propose and formalize an abstraction-based framework for computing explanations for inconsistency of ontology-enhanced KGs.
- We identify a fragment of OWL 2 for which inconsistency explanations can be computed by partitioning the input KG into small independent modules.
- For the identified OWL 2 fragment, we develop an efficient algorithm to compute all explanations for inconsistent knowledge graphs based on data abstraction.
- We evaluate our approach on NPD [30], Yago [31], and DBpedia [19] KGs with very promising results: we manage to compute thousands of inconsistency explanations and their patterns, for which computing directly on the original KGs is infeasible.

2 PRELIMINARIES

Knowledge Graphs and Ontologies. We assume countable pairwise disjoint sets $\mathsf{N_C}$, $\mathsf{N_P}$ and $\mathsf{N_I}$ of class names, e.g., `Company`, property names, e.g., `hasCustomer`, and individuals, e.g., `Toyota`. A *knowledge graph* (KG) $\mathcal{G}$ is a finite set of unary and binary facts, also called *assertions*, of the form $C(a)$ and $R(a, b)$, where $C \in \mathsf{N_C}$, $R \in \mathsf{N_P}$ and $a, b \in \mathsf{N_I}$, e.g., `PopularName(Nokia)` or `hasCustomer(McAfee,Nokia)`. We use $\mathsf{ind}(\mathcal{G})$ to denote the sets of individuals occurring in $\mathcal{G}$.

From class and property names one can (recursively) define complex *classes* C and *properties* P following the OWL 2 specification. In this work, we focus on the following types of classes and properties:

$P ::= R \mid$ `ObjectInverseOf(`P`)`

$C ::=$ `owl:Thing` $\mid$ `owl:Nothing` $\mid A \mid$ `ObjectComplementOf(`C`)` $\mid$ `ObjectIntersectionOf(`C_1, C_2`)` $\mid$ `ObjectUnionOf(`C_1, C_2`)` $\mid$ `ObjectSomeValuesFrom(`P`, owl:Thing)`,

Table 1: Syntax and semantics of the ontology language considered in this paper where A, R are a class name and property name, respectively; C and D are class expressions, P, S are property expressions, and a, b are individuals.

Syntax	Semantics
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
`ObjectInverseOf(`R`)`	$\{\langle e, d\rangle \mid \langle d, e\rangle \in R^{\mathcal{I}}\}$
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
`owl:Thing`	$\Delta^{\mathcal{I}}$
`owl:NoThing`	$\emptyset$
`ObjectComplementOf(`C`)`	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
`ObjectIntersectionOf(`C`)`	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
`ObjectUnionOf(`C, D`)`	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
`ObjectSomeValuesFrom(`P`, owl:Thing)`	$\{d \mid \exists e \in \Delta^{\mathcal{I}}. \langle d, e\rangle \in P^{\mathcal{I}}\}$
`SubClassOf(`C, D`)`	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
`SubObjectPropertyOf(`P, S`)`	$P^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
`TransitiveObjectProperty(`P`)`	$P^{\mathcal{I}} \circ P^{\mathcal{I}} \subseteq P^{\mathcal{I}}$
`ClassAssertion(`C, a`)`	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
`ObjectPropertyAssertion(`R, a, b`)`	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$

where $A \in \mathsf{N_C}$, $R \in \mathsf{N_P}$, C_1, C_2 are classes and P is a property. Classes and properties can be used to define axioms that formally describe the domain of interest, and the set of such axioms is called an *ontology*. Table 1 describes the axioms considered in this paper. Note that these axioms can express other common OWL 2 axioms such as: `InverseObjectProperties`, `ObjectPropertyDomain`, `ObjectPropertyRange`, `EquivalentClasses`, and `DisjointClasses`.

Inconsistency and Explanations. The semantics of knowledge graphs and ontologies is defined using the OWL 2 direct model-theoretic semantics via interpretations [24]. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and an *interpretation function* $\cdot^{\mathcal{I}}$, that assigns to each $A \in \mathsf{N_C}$ a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each $R \in \mathsf{N_R}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to each $a \in \mathsf{N_I}$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. This assignment is extended to (complex) classes and properties as shown in Table 1.

An interpretation $\mathcal{I}$ *satisfies* an axiom α (written $\mathcal{I} \models \alpha$) if the corresponding condition in Table 1 holds. Given a KG $\mathcal{G}$ and an ontology $\mathcal{O}$, $\mathcal{I}$ is a *model* of $\mathcal{G} \cup \mathcal{O}$ (written $\mathcal{I} \models \mathcal{G} \cup \mathcal{O}$) if $\mathcal{I} \models \alpha$ for all axioms $\alpha \in \mathcal{G} \cup \mathcal{O}$. We say that $\mathcal{G} \cup \mathcal{O}$ *entails* an axiom α (written $\mathcal{G} \cup \mathcal{O} \models \alpha$), if every model of $\mathcal{G} \cup \mathcal{O}$ satisfies α. A knowledge graph $\mathcal{G}$ is *inconsistent* w.r.t. an ontology $\mathcal{O}$ if no model for $\mathcal{G} \cup \mathcal{O}$ exists. In this case, we also say $\mathcal{G} \cup \mathcal{O}$ is inconsistent. Intuitively, $\mathcal{G} \cup \mathcal{O}$ is inconsistent when some facts of $\mathcal{G}$ contradict some axioms of $\mathcal{O}$.

An *explanation* for inconsistency of $\mathcal{G} \cup \mathcal{O}$ [13], denoted by $\mathcal{E} = \mathcal{E}_{\mathcal{G}} \cup \mathcal{E}_{\mathcal{O}}$ with $\mathcal{E}_{\mathcal{G}} \subseteq \mathcal{G}$ and $\mathcal{E}_{\mathcal{O}} \subseteq \mathcal{O}$, is a smallest subset-inclusion inconsistent subset of $\mathcal{G} \cup \mathcal{O}$. E.g., the KG part in Figure 1 stating that `McAffee` `hasCustomer` Toyota and Toyota `isCityOf` Japan together with the three example ontology axioms is an explanation. In general, inconsistency of $\mathcal{G} \cup \mathcal{O}$ may have multiple explanations.

Homomorphisms. We now define *homomorphisms* between KGs to establish the theoretical foundation for our approach.

Definition 2.1. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be knowledge graphs and $h\colon \mathsf{ind}(\mathcal{G}_1) \to \mathsf{ind}(\mathcal{G}_2)$ a mapping from the individuals occurring in $\mathcal{G}_1$ to individuals occurring in $\mathcal{G}_2$. We extend h to axioms in a straightforward way: $h(C(a)) = C(h(a))$, $h(R(a, b)) = R(h(a), h(b))$. We define $h(\mathcal{O}) = \mathcal{O}$ for every ontology $\mathcal{O}$ and define $h(\mathcal{G}_1) = \{h(\alpha) | \alpha \in \mathcal{G}_1\}$. We say that h is a *homomorphism* (from $\mathcal{G}_1$ to $\mathcal{G}_2$) if $h(\mathcal{G}_1) \subseteq \mathcal{G}_2$.

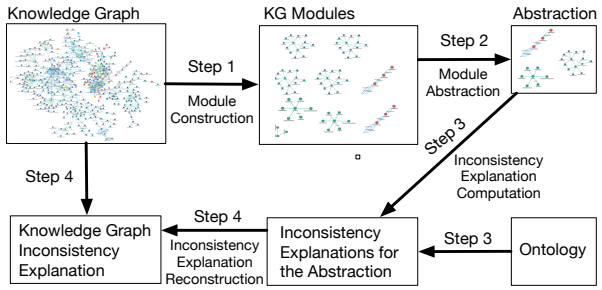

Figure 2: Approach overview.

The following well-known lemma states that entailment is preserved under homomorphisms.

Lemma 2.2. *Let $h\colon \mathsf{ind}(\mathcal{G}_1) \to \mathsf{ind}(\mathcal{G}_2)$ be a homomorphism between the knowledge graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. Then, for every ontology $\mathcal{O}$ and every axiom α, we have $\mathcal{G}_1 \cup \mathcal{O} \models \alpha$ implies $\mathcal{G}_2 \cup \mathcal{O} \models h(\alpha)$.*

3 EXPLAINING KG INCONSISTENCY

In this section, we present our approach for computing all explanations of KG inconsistency. We proceed by presenting the overview of the approach, and then describe the details of each step, followed by the proof of correctness. Finally, all steps are put together and presented in a complete detailed algorithm of the approach.

Approach Overview. In Figure 2, we present a high-level overview of our method. In Step 1, for every KG individual a we construct its module (Sec. 3.1). In Step 2, we detect commonalities across modules and compute their compact representations, i.e., abstractions (Sec. 3.2). The union of abstractions of all modules gives an abstraction for the whole KG. As shown in Sec. 4, this KG abstraction in practice is much smaller than the original KG. Step 3 takes as input the ontology and the computed abstractions and invokes an off-the-shelf reasoner to compute inconsistency explanations for the abstract modules, which are patterns for inconsistency explanations for the original KG. Finally, in Step 4 (Sec. 3.3), we reconstruct the explanations of the original KG from those computed in Step 3.

3.1 Modules and Inconsistency Locality Property

In KGs, entities are highly connected, and therefore, ontology axioms could trigger multi-hop reasoning steps, e.g., the transitivity axiom `TransitiveObjectProperty(foaf)`, where `foaf` stands for `friend of a friend`, combined with two facts `foaf(John,Mary)` and `foaf(Mary,Tom)` in a KG infers a new fact `foaf(John,Tom)`. However, under the considered ontology language, we discovered that the KG inconsistency has a locality property, i.e., the problem of checking inconsistency for a KG (w.r.t. an ontology $\mathcal{O}$) can be reduced to checking inconsistency for separated KG *modules* (w.r.t. $\mathcal{O}$).

Definition 3.1 (Modules). Given a KG $\mathcal{G}$ and an individual $a \in \mathsf{ind}(\mathcal{G})$, the *module* of a w.r.t. $\mathcal{G}$ is defined as $\mathcal{M}(a, \mathcal{G}) = \{\alpha \mid \alpha \in \mathcal{G}$ and a occurs in $\alpha\}$. We denote the set of all modules for individuals occurring in $\mathcal{G}$ as $\mathcal{M}_{\mathcal{G}} = \{\mathcal{M}(a, \mathcal{G}) \mid a \in \mathsf{ind}(\mathcal{G})\}$.

Intuitively, the module for an individual a in a knowledge graph is the set of all facts containing a. In our example in Figure 1, the module of `Nokia` is defined as $\mathcal{M}(\texttt{Nokia}, \mathcal{G}) = \{\texttt{PopularName(Nokia)}, \texttt{hasCustomer(McAffee, Nokia)}, \texttt{isCityOf(Nokia, Japan)}\}$.

Lemma 3.2. *Let $\mathcal{G}$ be a KG and $\mathcal{O}$ an ontology. Then $\mathcal{G} \cup \mathcal{O}$ is consistent iff $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$ is consistent for every $a \in \mathsf{ind}(\mathcal{G})$.*

Before proving Lemma 3.2 we show the following proposition.

Proposition 3.3. *Let $\mathcal{G}$ be a knowledge graph such that $\{R(a, b'), R(a', b)\} \subseteq \mathcal{G}$ for some property R and individuals a, b, a', b'. Furthermore, let $\mathcal{G}' = \mathcal{G} \cup \{R(a, b)\}$. Then, for every ontology $\mathcal{O}$, we have $\mathcal{G} \cup \mathcal{O}$ is consistent implies $\mathcal{G}' \cup \mathcal{O}$ is consistent.*

Proof of Proposition 3.3. We assume $\mathcal{G} \cup \mathcal{O}$ is consistent and show that $\mathcal{G}' \cup \mathcal{O}$ is consistent. The idea is to construct a model of $\mathcal{G}' \cup \mathcal{O}$ based on a model of $\mathcal{G} \cup \mathcal{O}$. Let $\mathcal{I}$ be a model of $\mathcal{G} \cup \mathcal{O}$. We construct an interpretation $\mathcal{J}$ as follows: $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$; $a^{\mathcal{J}} = a^{\mathcal{I}}$ and $A^{\mathcal{J}} = A^{\mathcal{I}}$ for every individual a and class name A, respectively; and for every property name P, we define: $P^{\mathcal{J}} = P^{\mathcal{I}} \cup \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \mid \mathcal{O} \models \texttt{SubObjectPropertyOf}(R, P)\} \cup \{\langle b^{\mathcal{I}}, a^{\mathcal{I}}\rangle \mid \mathcal{O} \models \texttt{SubObjectPropertyOf}(\texttt{ObjectInverseOf}(R), P)\}$.

We show $\mathcal{J}$ entails all axioms in $\mathcal{G} \cup \mathcal{O}$ except for the property transitivity axioms. Since $\mathcal{I} \models \mathcal{G} \cup \mathcal{O}$ and the interpretation of class names and individuals remains the same in $\mathcal{J}$, we have $\mathcal{J}$ entails each class assertion in $\mathcal{G}'$. Furthermore, from the definition of $\mathcal{J}$, the interpretations of properties are expanded such that $\langle a^{\mathcal{J}}, b^{\mathcal{J}}\rangle \in R^{\mathcal{J}}$, therefore, $\mathcal{J}$ entails each property assertion $R(a, b)$ in $\mathcal{G}'$. It remains to show that $\mathcal{J} \models \mathcal{O}$. We prove by induction that, for every class expression D, we have $D^{\mathcal{J}} = D^{\mathcal{I}}$.

- Cases $D = \texttt{owl:Thing}$, $D = \texttt{owl:NoThing}$, and $D = A$, where $A \in \mathsf{N_C}$ follow directly from the definition of $\mathcal{J}$.
- Case $D = \texttt{ObjectSomeValuesFrom}(R, \texttt{owl:Thing})$. We consider the case, where R is a property name; the case of inverse property is similar. Let d be an arbitrary element in $D^{\mathcal{J}}$. According to the semantics of existential axioms represented in Table 1, we have $d \in \texttt{ObjectSomeValuesFrom}(P, \texttt{owl:Thing})^{\mathcal{J}}$ iff there exists $e \in \Delta^{\mathcal{J}}$ s.t. $\langle d, e\rangle \in P^{\mathcal{J}}$. If $\langle d, e\rangle \in P^{\mathcal{I}}$, then $d \in D^{\mathcal{I}}$. Otherwise, from the definition of $\mathcal{J}$, $\langle d, e\rangle$ results from one of the cases in the role extension. We consider the case $d = a^{\mathcal{I}}, e = b^{\mathcal{I}}$, where $\mathcal{O} \models \texttt{SubObjectPropertyOf}(R, P)$ (the other cases are analogous). Since $R(a, b') \in \mathcal{G}$ and $\mathcal{I}$ is a model of $\mathcal{G} \cup \mathcal{O}$, we obtain $\langle a^{\mathcal{I}}, b'^{\mathcal{I}}\rangle \in P^{\mathcal{I}}$. Therefore, $d = a^{\mathcal{I}} \in D^{\mathcal{I}}$. Since d is arbitrary, we have $D^{\mathcal{J}} \subseteq D^{\mathcal{I}}$. Furthermore, since the interpretations of properties are expanded, we get $D^{\mathcal{I}} \subseteq D^{\mathcal{J}}$. Hence, $D^{\mathcal{I}} = D^{\mathcal{J}}$.
- Cases $D = \texttt{ObjectComplementOf}(D')$, $D = \texttt{ObjectIntersectionOf}(D_1, D_2)$, and $D = \texttt{ObjectunionOf}(D_1, D_2)$ follow from their semantics and the induction hypothesis.

For every subclass axiom $\texttt{SubClassOf}(C_1, C_2) \in \mathcal{O}$, we have $C_1^{\mathcal{J}} = C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}} = C_2^{\mathcal{J}}$, i.e. $\mathcal{J} \models \texttt{SubClassOf}(C_1, C_2)$. Since the interpretations of properties are expanded according to the sub property axioms, $\mathcal{J}$ entails every subproperty axiom in $\mathcal{O}$.

In the last step, $\mathcal{J}$ is extended to obtain $\mathcal{J}'$ by expanding the interpretations of properties such that $\mathcal{J}'$ entails every transitivity axiom in $\mathcal{O}$. One could repeat the previous steps to show that $\mathcal{J}'$ still entails other axioms in $\mathcal{O}$. Therefore, $\mathcal{J}' \models \mathcal{G}' \cup \mathcal{O}$, which implies that $\mathcal{G}' \cup \mathcal{O}$ is consistent. □

We are now ready to show the proof for Lemma 3.2.

Proof of Lemma 3.2. The "only-if direction" is trivial since each module of $\mathcal{G}$ is a subset of $\mathcal{G}$, and therefore, $\mathcal{G} \cup \mathcal{O}$ is consistent implies all modules are consistent. We prove the remaining direction by assuming that $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$ is consistent for all $a \in \text{ind}(\mathcal{G})$; we then show that $\mathcal{G} \cup \mathcal{O}$ is also consistent. This is done via the following steps. First, for each module, e.g., $\mathcal{M}(a, \mathcal{G})$ of $\mathcal{G}$, we rename all individuals in it except for a s.t. individuals occurring in each module are pairwise disjoint. It allows us to use the disjoint union of the models to construct an interpretation which is close to a model of $\mathcal{G} \cup \mathcal{O}$. For each module $\mathcal{M}(a, \mathcal{G})$, we denote the resulting renamed module as $\mathcal{M}'(a, \mathcal{G})$, and since $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$ is consistent, we have $\mathcal{M}'(a, \mathcal{G}) \cup \mathcal{O}$ is consistent (by Lemma 2.2). Let $\mathcal{I}_a$ be a model of $\mathcal{M}'(a, \mathcal{G}) \cup \mathcal{O}$ for each individual $a \in \text{ind}(\mathcal{G})$. We define an interpretation $\mathcal{I}$ as the disjoint union of all $\mathcal{I}_a$ for all individuals a occurring in $\mathcal{G}$. Then, $\mathcal{I} \models \{\mathcal{M}'(a, \mathcal{G}) \mid a \in \text{ind}(\mathcal{G})\} \cup \mathcal{O}$, which implies that $\mathcal{G}_1 \cup \mathcal{O}$, where $\mathcal{G}_1 = \bigcup_{a \in \text{ind}(\mathcal{G})} \mathcal{M}'(a, \mathcal{G})$, is consistent. Second, we extend $\mathcal{G}_1$ by adding all property assertions in $\mathcal{G}$ to obtain $\mathcal{G}_2$ defined as $\mathcal{G}_2 = \mathcal{G}_1 \cup \{\texttt{ObjectPropertyAssertion}(R, a, b) \mid \texttt{ObjectPropertyAssertion}(R, a, b) \in \mathcal{G}_1\}$. By Proposition 3.3, we have $\mathcal{G}_2 \cup \mathcal{O}$ is consistent. Finally, by monotonicity, we obtain $\mathcal{G} \cup \mathcal{O}$ is consistent because $\mathcal{G} \subseteq \mathcal{G}_2$ and $\mathcal{G}_2 \cup \mathcal{O}$ is consistent. □

Using Lemma 3.2, we can show that all explanations for inconsistency of a KG can be obtained by computing explanations for its modules and thus in Step 1 of our approach we construct modules.

Theorem 3.4. *Let $\mathcal{G}$ be a KG and $\mathcal{O}$ an ontology s.t. $\mathcal{G} \cup \mathcal{O}$ is inconsistent. Then, $\mathcal{E}$ is an explanation for inconsistency of $\mathcal{G} \cup \mathcal{O}$ iff there exists $a \in \text{ind}(\mathcal{G})$ s.t. $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$ is inconsistent and $\mathcal{E}$ is an explanation for the inconsistency of $\mathcal{M}(a, \mathcal{G}) \cup \mathcal{O}$.*

Proof of Theorem 3.4. The "if" direction of Theorem 3.4 holds as, by monotonicity, all explanations for inconsistency of the modules of $\mathcal{G}$ are explanations for the inconsistency of $\mathcal{G}$. We now show the "only-if" direction. Let $\mathcal{E} = \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$, where $\mathcal{E}_\mathcal{G} \subseteq \mathcal{G}$ and $\mathcal{E}_\mathcal{O} \subseteq \mathcal{O}$, is an inconsistency explanation for $\mathcal{G} \cup \mathcal{O}$. Since $\mathcal{E}$ is inconsistent, by Lemma 3.2, an individual a occurring in $\mathcal{G}_\mathcal{E}$ exists, such that $\mathcal{M}(a, \mathcal{E}_\mathcal{G}) \cup \mathcal{O}_\mathcal{E}$ is inconsistent. Let $\mathcal{E}'$ be an explanation for inconsistency of $\mathcal{M}(a, \mathcal{E}_\mathcal{G}) \cup \mathcal{E}_\mathcal{O}$, which is also an explanation for $\mathcal{G} \cup \mathcal{O}$. Since $\mathcal{E}' \subseteq \mathcal{M}(a, \mathcal{E}_\mathcal{G}) \cup \mathcal{E}_\mathcal{O} \subseteq \mathcal{E}$ and $\mathcal{E}$ is minimal, we have $\mathcal{E} = \mathcal{E}'$, which is what is required to be proved. □

Using Theorem 3.4, one can already compute inconsistency explanations for a KG based on its modules, even without further steps. We show in Section 4 that this gives a significant speed-up over the baseline. More importantly, we observe that many modules have similar structures and computing inconsistency explanations for multiple modules can be performed on just one representative module. This motivates the further steps for KG abstraction.

3.2 Graph Abstraction

Let $\mathsf{N_V}$ be a countable set of *abstract* individuals that is disjoint with $\mathsf{N_C}$, $\mathsf{N_P}$, and $\mathsf{N_I}$. An *abstract KG* is a KG consisting of facts over abstract individuals, i.e., a finite set of facts of the form $C(u)$, $R(u, v)$, where $u, v \in \mathsf{N_V}, C \in \mathsf{N_C}, R \in \mathsf{N_P}$. Intuitively, each abstract individual is a representative for several ones in the original KG.

Given an ontology $\mathcal{O}$ and a KG $\mathcal{G}$, an abstract graph $\mathcal{G}'$ is *inconsistency preserving* (for $\mathcal{G}$) when $\mathcal{G}'$ is consistent (w.r.t. $\mathcal{O}$) iff $\mathcal{G}$ is consistent. An abstract graph $\mathcal{G}'$ is *explanation preserving* (for $\mathcal{G}$) if for every inconsistency explanation $\mathcal{E}$ of $\mathcal{G}$, there exists an inconsistency explanation $\mathcal{E}'$ of $\mathcal{G}'$ such that $\mathcal{E}'$ can be homomorphically mapped to $\mathcal{E}$. An inconsistency and explanation preserving abstract graph $\mathcal{G}'$ (for $\mathcal{G}$) is called an *abstraction* of $\mathcal{G}$. Clearly, an abstraction of any KG satisfies the requirements R1–R2, but it does not always guarantee to satisfy R3, which is important for handling large KGs. In the following, we propose a method for constructing KG abstractions that are significantly smaller than the original real-world KGs, i.e., R3 holds for them in practice (see Section 4).

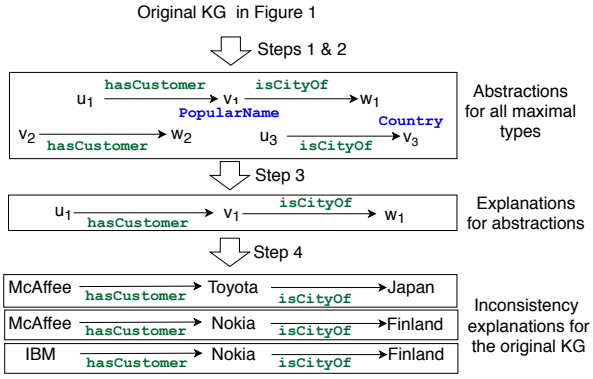

Figure 3: KG abstraction and explanations (without $\mathcal{E}_\mathcal{O}$).

The abstraction of a KG is computed by identifying modules with similar structures and then representing each group of similar modules by just a single representative one. To characterize the structure of modules, we rely on the *local type* of individuals [9, 11]. Intuitively, a local type of an individual represents a set of classes and the incoming and outgoing properties occurring in the corresponding module for that individual.

Definition 3.5 (Local Types). Let $\mathcal{G}$ be a KG and $a \in \text{ind}(\mathcal{G})$, then, the *local type* of a w.r.t. $\mathcal{G}$, written as $\tau(a, \mathcal{G})$ or $\tau(a)$ when $\mathcal{G}$ is clear from the context, is defined as a tuple $\tau(a) = \langle \tau_i(a), \tau_c(a), \tau_o(a) \rangle$, where $\tau_i(a) = \{R \mid R(c, a) \in \mathcal{G}\}$, $\tau_c(a) = \{A \mid A(a) \in \mathcal{G}\}$, and $\tau_o(a) = \{S \mid S(a, b) \in \mathcal{G}\}$.The local type $t = \langle t_i, t_c, t_o \rangle$ is *smaller or equal* than the local type $t' = \langle t'_i, t'_c, t'_o \rangle$, written as $t \preceq t'$, iff $t_i \subseteq t'_i, t_c \subseteq t'_c$, and $t_o \subseteq t'_o$. We write $t \prec t'$ iff $t \preceq t'$ and $t \neq t'$. Given a set $\mathcal{S}$ of local types, a local type t is called *maximal* (in $\mathcal{S}$) if $t \in \mathcal{S}$ and no $t' \in \mathcal{S}$ exists s.t. $t \prec t'$.

Each local type consists of sets of classes and properties. We now define a *(star shape) abstraction* for each local type by instantiating these classes and properties with abstract individuals in $\mathsf{N_V}$.

Definition 3.6 (Star Shape Abstractions). Let $t = \langle t_i, t_c, t_o \rangle$ be a local type, the *star shape abstraction* for t is defined as $\text{abs}(t) = \{A(v_t) \mid A \in t_c\} \cup \{R(u^R_t, v_t) \mid R \in t_i\} \cup \{S(v_t, w^S_t) \mid S \in t_o\}$, where v_t, u^R_t, w^S_t are unique *abstract individuals* for each t, R, S, and v_t.

For simplicity, we will omit the subscripts and superscripts of individuals in star shape abstractions. These are only needed to formally make those individuals distinguishable.

Example 3.7. `Toyota` and `Nokia` in Figure 1 have the same local type defined as $t = \langle \{\texttt{PopularName}\}, \{\texttt{hasCustomer}\}, \{\texttt{isCityOf}\} \rangle$. For t, $\text{abs}(t) = \{\texttt{PopularName}(v), \texttt{hasCustomer}(u, v), \texttt{isCityOf(v, w)}\}$ is the star shape abstraction.

Algorithm 1: Computing explanations for inconsistency of a knowledge graph $\mathcal{G}$ w.r.t. an ontology $\mathcal{O}$

Input : A knowledge graph $\mathcal{G}$ and an ontology $\mathcal{O}$
Output: The set `allExpls` of all explanations for inconsistency of $\mathcal{G} \cup \mathcal{O}$

```
1 allExpls ← ∅
  /* Step 1 & 2: compute local types of all individuals in G */
2 types ← {τ(a, G) | a ∈ ind(G)}
3 foreach maximal τ ∈ types do
      /* Step 3: compute explanations for the star shape
         abstraction of τ using a reasoner */
4     X ← all explanations for inconsistency of abs(τ) ∪ O
      /* Step 4: obtain the explanations for G */
5     foreach E = E_G ∪ E_O ∈ X do
          /* compute the local type of v_τ in E_G */
6         τ' = τ(v_τ, E_G)
7         newExpls ← all realizations of τ' in G
8         allExpls ← allExpls ∪ newExpls
9 return allExpls
```

For each explanation $\mathcal{E} = \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{O}$, $\mathcal{E}_\mathcal{G}$ and $\mathcal{E}_\mathcal{O}$ are minimal. Therefore, we only need to consider maximal local types. One can show that for τ and τ' such that $\tau \prec \tau'$, for every ontology $\mathcal{O}$, if $\text{abs}(\tau) \cup \mathcal{O}$ is inconsistent then $\text{abs}(\tau') \cup \mathcal{O}$ is also inconsistent and inconsistency explanations of $\text{abs}(\tau') \cup \mathcal{O}$ can be used to obtain all those of $\text{abs}(\tau) \cup \mathcal{O}$. Thus, smaller types are irrelevant for computing explanations.

3.3 Inconsistency Explanation Reconstruction

We now define the *realization of a local type* that allows us to reconstruct explanations for the original KG from those for abstractions.

Definition 3.8 (Realization of a local type). Let $\mathcal{G}$ be a KG and τ' a local type. A *realization* of τ' for an individual $a \in \text{ind}(\mathcal{G})$ is an inclusion-smallest subset $\text{real}_{a,\tau'}$ of $\mathcal{G}$ s.t. $\tau(a, \text{real}_{a,\tau'}) = \tau'$. The *realization* of τ' in $\mathcal{G}$ is the set of all realizations of τ' for each individual occurring in $\mathcal{G}$.

In Figure 3, we present the KG-part $\mathcal{E}_\mathcal{G}$ of an inconsistency explanation for the abstractions and three corresponding explanations (i.e., the realizations of $\tau(v_1, \mathcal{E}_\mathcal{G})$), for the original KG in Figure 1. The former is regarded as *explanation pattern* for the latter.

3.4 Bringing all Together

Finally, we proceed with Algorithm 1 that computes all explanations for KG $\mathcal{G}$ w.r.t. an ontology $\mathcal{O}$ using abstraction. Since it iterates over local types, whose number is bounded by $|\text{ind}(\mathcal{G})|$, and the number of explanations for abstractions as well as the number of realizations for types are bounded by the signature of $\mathcal{G}$ and $\mathcal{O}$, Algorithm 1 terminates. Figure 3 illustrates all steps of the algorithm for the example in Figure 1. The following theorem shows its correctness.

Theorem 3.9. *Given a KG $\mathcal{G}$ and an ontology $\mathcal{O}$ as inputs for Alg. 1, $\cup_{a \in \text{ind}(\mathcal{G})} \text{abs}(\tau(a))$ is an abstraction of $\mathcal{G}$ and the set* `allExpls` *consists of all inconsistency explanations for $\mathcal{G}$ w.r.t. $\mathcal{O}$.*

Proof of Theorem 3.9. The abstraction $\text{abs}(\tau)$ for every type τ can be homomorphically mapped to any realization of τ in $\mathcal{G}$. Therefore, by Lemma 2.2, if τ' is the local type of some individual v_τ in an inconsistency explanation of $\text{abs}(\tau) \cup \mathcal{O}$, then the realizations of τ' (together with ontology axioms) in $\mathcal{G}$ are inconsistency explanations of $\mathcal{G} \cup \mathcal{O}$. This shows that every element of `allExpls` (in Algorithm 1) is an inconsistency explanation of $\mathcal{G} \cup \mathcal{O}$.

To prove that `allExpls` contains all explanations of $\mathcal{G} \cup \mathcal{O}$, we rely on the claim that $\mathcal{G} \cup \mathcal{O}$ is consistent iff $\text{abs}(\tau(a)) \cup \mathcal{O}$ is consistent for every individual a in $\mathcal{G}$, which can be proved based on Proposition 3.3 as done in the proof of Lemma 3.2. Then, similar to the proof of Theorem 3.4, we show that every explanation for inconsistency of $\mathcal{G} \cup \mathcal{O}$ can be obtained via some inconsistency explanation of $\text{abs}(\tau) \cup \mathcal{O}$ for some maximal local type τ. □

Table 2: Statistics of the datasets.

KG	Axioms	Assertions	Entities	Classes	Relations
NPD	678	929,710	264,081	343	142
Yago	1,045	7,321,308	3,275,593	960	38
DBpedia	4,287	22,955,173	5,867,913	685	663

Table 3: Results of inconsistency explanation computation.

	Method	Total Modules	Processed Modules	Patterns	Explanations	Time (hours)
NPD	*Pellet*	1	0	-	-	72
	Modular	264,081	243,467	-	41,128	72
	Abstraction	11,970	11,866	20,251	60,554	4.5
Yago	*Pellet*	1	0	-	-	72
	Modular	3,275,593	1,624,196	-	1,547	72
	Abstraction	2,821	2,821	69	3,565	0.1
DBpedia	*Pellet*	1	0	-	-	72
	Modular	5,867,913	647,443	-	73,152	72
	Abstraction	91,084	90446	1,797	20,093,617	21

4 EXPERIMENTS

We have implemented our method for computing inconsistency explanations in a system prototype and evaluated it on three real world KGs equipped with their ontologies: the Norwegian Petroleum Directorate (NPD) [30], Yago [31], and DBpedia [19]. Since the original version of the NPD dataset is consistent, for the purpose of our experiments we introduced inconsistency by adding 2% of random property and class assertions. We used the ontologies provided with NPD and DBpedia KGs. For Yago, we took the union of the available simple taxonomy and its relational schema as the ontology. More details of our experiments are publicly available.[1]

Experiment Setup. To demonstrate the benefits of our method (*Abstraction*) presented in Alg. 1, we compare it with respect to the number of detected inconsistency explanations against the following baselines: module-based implementation as described in Section 3.1 (*Modular*) and the off-the-shelf reasoner [29] (*Pellet*), in which a KG is processed as a whole. Note that for computing explanations, Pellet is as efficient as other state-of-the-art reasoners, e.g., HermiT [8]. Table 2 presents the sizes of the test ontologies and of the KGs, and the number of entities, classes and properties.

All experiments were performed on a 48 cores machine with 500GB of memory. Pellet reasoner was invoked for computing inconsistency explanations in all settings. In the *Abstraction* setting, we regard each abstraction generated for a maximal local type (described in Algorithm 1) as a module. We set a timeout of 72 hours for the overall computation and 2 minutes for processing every module in the *Abstraction* and *Modular* settings.

[1] https://github.com/boschresearch/kg_inconsistency_explanation

Table 4: *Top* frequent inconsistency explanation patterns.

	Abstract KG Assertions	Ontology Axioms	Examples	#Explanations
Yago	wn:building(v) isLocatedIn(u,v)	DisjointClasses(wn:building, GeoEntity) ObjectPropertyRange(isLocatedIn,GeoEntity)	wn:building(<J._Forrestal_Building>) isLocatedIn(<NNSA>,<J._Forrestal_Building>)	1,768
DBpedia	bandMember(u_1,v) instrument(u_2,v)	SubClassOf(Ship,MeanOfTransportation) SubClassOf(Instrument,Product), SubClassOf(Product,Ship) DisjointClasses(MeanOfTransportation,Person) ObjectPropertyRange(bandMember,Person) ObjectPropertyRange(instrument,Instrument)	instrument(<African_blues>,<Djembe>) bandMember(<Baka_Beyond>,<Djembe>)	11,357,430

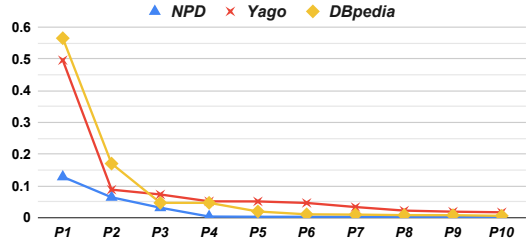

Figure 4: Distribution of the top-10 frequent explanations.

Results. In Table 3 , we compare the three methods with respect to the total number of modules computed, the number of modules processed within the timeout and the number of found explanation patterns together with the actual explanations that they yield.

Pellet reasoner failed to process all of the test KGs within the given timeout, i.e., no inconsistency explanations have been computed by this baseline method. Naively splitting the KG into smaller modules is already beneficial yielding some explanations. However, as the number of modules to be considered is still very large, many of them cannot be processed within the time limit, and hence, still only few inconsistency explanations are detected.

The *Abstraction* method significantly reduces the number of modules to be considered to 0.1% , 1.55%, and 4.5% of the original modules (*i.e.* computed by the *Modular* method) for Yago, DBpedia and NPD respectively. Although not all of the modules could be handled by the invoked reasoner, we managed to process a much larger fraction of the KGs compared to the baselines.

Inconsistency explanations were computed by our method within 6 minutes, 4.5 hours, and 21 hours for Yago, NPD and DBpedia respectively, which naturally correlates with the size and complexity of the considered KGs and their ontologies. The results demonstrate that our method can process large KGs within reasonable times, which are beyond of what earlier proposed approaches can handle.

Figure 4 depicts the *top-10* most frequent explanation patterns (*X-axis*), and the percentage of concrete inconsistency explanations they represent (*Y-axis*). The majority of explanations follow one of the 5 prominent patterns. Importantly, the analysis of these patterns may reveal systematic issues in the information extraction process, which makes the patterns valuable on their own. As shown in Table 4, about 50% of explanations result from the fact that the *ontology* specifies that the range of `isLocatedIn` is `GeoEntity`, which is disjoint with `building`. However, in the KG, there are 1.7K triples connecting some entity to an entity belonging to the `building` class via the `isLocatedIn` relation. Similarly, in DBpedia, about 57% of explanations are caused by an entity being an object of triples with relations `bandMember` and `playInstrument`, whose domains are `Person` and `Instrument` respectively, which are known to be disjoint.

5 RELATED WORK

The problem of detecting erroneous information in KGs has recently gained a lot of attention. Proposed approaches can be roughly divided into statistics-based [12, 20–22, 27], logic-based [1, 7, 17, 18, 37], and the combination of the first two groups [21, 32], where constraints are learned (e.g., from edit history [32]) and exploited for inconsistency detection. Among the statistics-based methods, the closest to ours are [22, 27], where faulty triples are identified by exploiting statistical distributions of KG relations and types. Another relevant work from this category [15] introduces a machine learning model to measure the trustworthiness of triples.

Our method falls into the logic-based group, in which constraints [25] or ontologies [2, 16] are utilized to identify and repair errors in KGs. In particular, the works [1, 13, 33] are the most relevant; however, in contrast to our method they do not exploit module-based splitting of the KG and the abstractions for inconsistency explanation computation, and do not scale as well as our method.

The idea of using compressed/summarized data and modules to address the scalability issue in large KGs has been explored in several works [4, 35]. Our method is similar in spirit to the abstraction refinement approach [9–11] developed for the purpose of KG expansion, which is different from our goal of computing inconsistency explanations. Importantly, we have identified a new fragment of OWL in which the refinement phase is not necessary and have shown that only maximal local types need to be considered for computing explanations. Computing inconsistency explanations is also done via caching [28], but only limited types of explanations can be discovered, e.g., the most frequent explanations reported in Section 4 have not been detected by the method from [28].

6 CONCLUSIONS

We presented a novel method for computing inconsistency explanations of KGs based on KG abstraction. The applicability of our explanations is manifold. For instance, they can be used to facilitate error resolution in KGs e.g., following the automatic approach of Bienvenu *et al.* [1] or that of Tanon *et al.* [32], where a human decides which axioms are to be deleted. Another advantage of our method is that it provides error patterns which could reveal systematic issues in the KG construction process. We plan to exploit this direction in the future e.g., by exploiting inconsistency explanations to guide the information extraction process.

Another future direction is to extend our method to handle more expressive ontologies. Currently, it is restricted to the presented OWL 2 fragment. Adding other standard OWL 2 axioms such as `sameAs` or unrestricted form of `someValuesFrom/allValuesFrom` axioms, would make the approach incomplete. A possibility for extension of our algorithm to larger fragments of OWL 2 is by adapting it

in a "pay-as-you-go" fashion, e.g., by performing additional reasoning steps to handle the new axioms and employing the refinement phase as done in [11].

REFERENCES

[1] Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. 2016. Query-Driven Repairing of Inconsistent DL-Lite Knowledge Bases. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016.* 957–964.

[2] Stefan Bischof, Markus Krötzsch, Axel Polleres, and Sebastian Rudolph. 2014. Schema-Agnostic Query Rewriting in SPARQL 1.1. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I.* 584–600.

[3] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008.* 1247–1250.

[4] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg, and Kavitha Srinivas. 2009. Scalable highly expressive reasoner (SHER). *J. Web Semantics* (2009).

[5] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014.* 601–610.

[6] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31, 3 (2010), 59–79.

[7] Mohamed H. Gad-Elrab, Daria Stepanova, Jacopo Urbani, and Gerhard Weikum. 2019. ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19).* 87–95.

[8] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. 2014. HermiT: An OWL 2 Reasoner. *J. Autom. Reasoning* 53, 3 (2014), 245–269.

[9] Birte Glimm, Yevgeny Kazakov, Thorsten Liebig, Trung-Kien Tran, and Vincent Vialard. 2014. Abstraction Refinement for Ontology Materialization. In *The Semantic Web – ISWC 2014.* Springer, 180–195.

[10] Birte Glimm, Yevgeny Kazakov, and Trung-Kien Tran. 2016. Scalable Reasoning by Abstraction Beyond DL-Lite. In *Web Reasoning and Rule Systems.* Springer, 77–93.

[11] Birte Glimm, Yevgeny Kazakov, and Trung-Kien Tran. 2017. Ontology Materialization by Abstraction Refinement in Horn SHOIF. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (San Francisco, California, USA) *(AAAI'17).* AAAI Press, 1114–1120.

[12] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016.* 327–336.

[13] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2009. Explaining Inconsistencies in OWL Ontologies. In *Scalable Uncertainty Management.* Springer, Berlin, Heidelberg, 124–137.

[14] Ian Horrocks. 2008. Ontologies and the semantic web. *Commun. ACM* 51, 12 (2008), 58–67.

[15] Shengbin Jia, Yang Xiang, Xiaojun Chen, Kun Wang, and Shijia E. 2019. Triple Trustworthiness Measurement for Knowledge Graph. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019.* 2865–2871.

[16] Holger Knublauch and Dimitris Kontokostas. 2017. *Shapes constraint language (SHACL).* Technical Report. W3C. https://www.w3.org/TR/shacl/

[17] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven evaluation of linked data quality. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014.* 747–758.

[18] Freddy Lécué and Jeff Z. Pan. 2015. Consistent Knowledge Discovery from Evolving Ontologies. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* 189–195.

[19] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6 (2015), 167–195.

[20] Jiaqing Liang, Yanghua Xiao, Yi Zhang, Seung-won Hwang, and Haixun Wang. 2017. Graph-Based Wrong IsA Relation Detection in a Large-Scale Lexical Taxonomy. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 1178–1184.

[21] Yanfang Ma, Huan Gao, Tianxing Wu, and Guilin Qi. 2014. Learning Disjointness Axioms With Association Rule Mining and Its Application to Inconsistency Detection of Linked Data. In *The Semantic Web and Web Science - 8th Chinese Conference, CSWS 2014, Wuhan, China, August 8-12.* 29–41.

[22] André Melo and Heiko Paulheim. 2017. Detection of Relation Assertion Errors in Knowledge Graphs. In *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017.* 22:1–22:8.

[23] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-Ending Learning. In *Proc. of AAAI.* 2302–2310.

[24] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. 2012. *OWL 2 Web Ontology Language Direct Semantics (Second Edition).* Technical Report. https://www.w3.org/TR/owl-direct-semantics/

[25] Ndapandula Nakashole, Mauro Sozio, Fabian M. Suchanek, and Martin Theobald. 2012. Query-Time Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules. In *Proceedings of the Second International Workshop on Searching and Integrating New Web Data Sources, Istanbul, Turkey, August 31, 2012.* 15–20.

[26] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8, 3 (2017), 489–508.

[27] Heiko Paulheim and Christian Bizer. 2014. Improving the Quality of Linked Data Using Statistical Distributions. *Int. J. Semantic Web Information Systems* 10, 2 (2014), 63–86.

[28] Heiko Paulheim and Aldo Gangemi. 2015. Serving DBpedia with DOLCE – More than Just Adding a Cherry on Top. In *The Semantic Web - ISWC 2015.* Springer, 180–196.

[29] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. 2007. Pellet: A practical OWL-DL reasoner. *J. Web Semant.* 5, 2 (2007).

[30] Martin G. Skjæveland, Espen H. Lian, and Ian Horrocks. 2013. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In *The Semantic Web – ISWC 2013*, Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz (Eds.). 162–177.

[31] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. of WWW.* 697–706.

[32] Thomas Pellissier Tanon, Camille Bourgaux, and Fabian M. Suchanek. 2019. Learning How to Correct a Knowledge Base from the Edit History. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019.* 1465–1475.

[33] Andrei Voronkov. 2006. Inconsistencies in Ontologies. In *JELIA (Lecture Notes in Computer Science)*, Vol. 4160. Springer, 19.

[34] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledge base. *CACM* 57, 10 (2014), 78–85.

[35] Sebastian Wandelt and Ralf Möller. 2012. Towards ABox Modularization of semi-expressive Description Logics. *Applied Ontology* 7, 2 (2012).

[36] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Trans. Knowl. Data Eng.* 29, 12 (2017), 2724–2743.

[37] Jiewen Wu and Freddy Lécué. 2014. Towards Consistency Checking over Evolving Ontologies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014.* 909–918.