Probably Approximately Correct Completion of Description Logic Knowledge Bases

Sergei Obiedkov[1], Barış Sertkaya[2], and Denis Zolotukhin[1]

[1] National Research University Higher School of Economics, Moscow, Russia
sergei.obj@gmail.com, ddzolotukhin@edu.hse.ru,
[2] Frankfurt University of Applied Sciences, Germany
sertkaya@fb2.fra-uas.de

Abstract. We propose an approach for approximately completing a TBox w.r.t. a fixed model. By asking implication questions to a domain expert, our method approximates the subsumption relationships that hold in expert's model and enriches the TBox with the newly discovered relationships between a given set of concept names. Our approach is based on Angluin's exact learning framework and on the attribute exploration method from Formal Concept Analysis. It brings together the best of both approaches to ask only polynomially many questions to the domain expert.

1 Introduction

Ontology development is an error-prone and time consuming task that is faced in various application domains. As the number and the size of the ontologies used in practice grow, methods for maintaining their quality become more and more important. Several methods have been proposed to this purpose. Detecting inconsistencies and inferring new consequences have been of major interest since the early days of the DL-research [4, 6]. There are also promising approaches that help to pinpoint the axioms in a knowledge base that cause inconsistency or other unwanted consequences [20, 11, 21, 16, 18, 17, 19]. These approaches address the quality dimension of soundness of an ontology. In [5], the other quality dimension, namely completeness of an ontology was addressed.

There the problem of completing a DL knowledge base w.r.t. an intended model has been considered. The aim of the presented approach there is to capture all relationships between a fixed set of interesting concepts that hold in the intended model of a domain expert. The notion of completeness that was used is the following:

– If an implication holds in the intended model, then it should follow from the TBox, and
– if it does not hold in this model, then the ABox should contain a counterexample to this implication.

To this purpose, a method from Formal Concept Analysis (FCA) [10] called attribute exploration was employed. Attribute exploration [8] is an interactive knowledge acquisition method that acquires complete knowledge about an application domain via querying an expert of this domain. The queries are of type "*Do all objects that have attributes* $a_1, a_2, \ldots, a_n$ *also have attributes* $b_1, b_2, \ldots, b_m$ *?*". If the expert confirms such a query, then an implication has been found that was missing and the current knowledge is extended with this new implication. If the expert rejects the query, then he is asked to provide a counterexample, i.e., an object that has all the attributes $a_1, a_2, \ldots, a_n$, but does not have at least one of the attributes $b_1, b_2, \ldots, b_m$. In this case, the knowledge is extended with the new object. The method terminates when all such queries are answered.

However, the downside of this method is that, in the worst case, the number of queries issued to the expert can be exponential not only in the number of the attributes, but also in the size of the smallest logically complete set of valid implications (implication basis) [9]. In order to overcome this problem, a probably approximately correct (PAC) version of the attribute exploration algorithm based on Angluin's exact learning framework [1, 2] has been proposed in [7, 15]. This version of the algorithm computes an approximation of the implication basis of the domain being learnt by issuing only polynomially many queries to the domain expert. The queries used there are implication queries just like in the classical attribute exploration algorithm.

In the present work, we apply the approach developed in [7] in the context of knowledge base completion. We use this approach for approximately completing a knowledge base w.r.t. a fixed model, which represents the expert's view of the application domain. Our aim is to approximate this view via issuing only polynomially many implication queries to the expert and enriching the knowledge base with the implications that are discovered. The resulting knowledge base is, with a specified probability, an approximation of this view within a specified error bound. Most of our results easily transfer from [7].

More precisely, our setting is the following. The domain expert has perfect knowledge of the application domain and has a TBox representing this domain, which is possibly incomplete, i.e., there are some subsumption relationships that hold in the expert's model but do not follow from the TBox. The domain expert is not able to formulate these subsumption relationships, but he is able to answer questions of the form "*Are all instances of the concepts* $C_1 \ldots C_n$ *also instances of the concepts* $D_1 \ldots D_m$ *?*" with "*yes*" or "*no*". In such a setting, our aim is to compute an approximation of the expert's model and complete the TBox with the missing subsumption relationships we have detected. The quality measure of our approximation is defined as in [7]. It is based on the difference between the set of models of the implications detected so far and the set of models of the actual implications that hold in the application domain. Note that, as the approach from [5], the approach presented here applies to an arbitrary description logic, provided it allows for the conjunction and negation constructors.

Angluin's exact learning framework has already been employed in the context of DLs in [13, 14]. There the authors investigate the computational complexity of exact learning of lightweight DL ontologies. They identify the fragments of $\mathcal{EL}$ and DL-Lite that allow for learnability of ontologies formulated in these fragments with polynomially many queries to a domain expert. Our approach is based on the same framework with a different aim. We want to approximately learn the implications holding in the expert's model using only implication queries. Giving up on exactness allows us to do this issuing only polynomially many queries.

We proceed as follows. In Section 2, we introduce the basic notions related to implications and the exact learning framework. In Section 3, we provide a PAC learning algorithm that computes an approximation of the subsumption relationships that hold in the model of a domain expert. In Section 4 we modify this algorithm to compute an approximate completion of a TBox.

2 Preliminaries

In Angluin's exact learning framework [1], an algorithm for learning a Horn formula via querying an oracle has been presented [2]. This algorithm issues two types of queries to the oracle, namely *membership queries*, which ask whether a specific truth assignment is a model of the formula being learnt, and *equivalence queries*, which ask whether the hypothesis is logically equivalent to the formula that is being learnt. A negative answer to an equivalence query is supported by a *counterexample*, either *positive* (a model of the target formula contradicting the hypothesis), or *negative* (an assignment satisfying the hypothesis, but not the target formula). It has been shown that, in the presence of these two types of queries, this algorithm learns the target formula with only polynomially many queries.

It is easy to see that exact learning of a Horn formula with polynomial number of queries is not possible if only membership queries are allowed. However, in real-world applications, finding domain experts that can answer equivalence queries is very unlikely because such an expert should be able to produce a negative counterexample if the equivalence query does not hold. In order to overcome this difficulty and still issue only polynomially many queries to the domain expert, an approach for approximating a set of implications using only implication queries has been presented in [7]. It is based on the idea of transforming an exact learning algorithm with equivalence queries into a probably approximately correct (PAC) algorithm without equivalence queries presented in [2]. In this approach, a random sampling strategy to search for a counterexample is used instead of relying on equivalence queries for obtaining such counterexamples.

In the context of FCA, propositional variables are called attributes and Horn formulas are called implications. In the following, these notions will sometimes be used interchangeably.

Definition 1. *Let M be a set of attributes and $A \to B$ be an implication over M with $A \subseteq M$ and $B \subseteq M$ or $B = \bot$. We say that a set $X \subseteq M$* models *$A \to B$ (is its* model*) if $A \not\subseteq X$ or $B \subseteq X$. We denote it as $X \models A \to B$. We say that X* models *a set of implications $\mathcal{L}$ over M if X models every implication in $\mathcal{L}$.*

In terms of propositional logic, one would say that X is a truth assignment and $A \to B$ is a Horn formula. Note that, according to Definition 1, X is a model of $A \to \bot$ if and only if $A \nsubseteq X$.

Definition 2. *For a set of implications $\mathcal{L}$ over M and a set $X \subseteq M$, the* implicational closure *of X under $\mathcal{L}$, denoted by $\mathcal{L}(X)$ is the smallest $Y \subseteq M$ such that*

- *$X \subseteq Y$, and*
- *$A \to B \in \mathcal{L}$ and $A \subseteq Y$ imply $\bot \neq B \subseteq Y$.*

If no such Y exists, then we say that the closure of X is $\bot$. We also say that X is closed *under $\mathcal{L}$ if $X = \mathcal{L}(X)$.*

Next we transfer this terminology to the DL context. For basic notions and notation of description logics, please refer to [4].

Definition 3. *For a given finite set of concept descriptions M and an implication $A \to B$ over M, we say that $A \to B$* holds *in $\mathcal{I}$ if $(\sqcap A)^{\mathcal{I}} \subseteq (\sqcap B)^{\mathcal{I}}$ for $B \subseteq M$ or $(\sqcap A)^{\mathcal{I}} = \varnothing$ for $B = \bot$.*

Here and in the following, $\sqcap A$ stands for the conjunction of concept descriptions from A.

Definition 4. *Let $\mathcal{T}$ be a consistent TBox, M be a finite set of concept descriptions, and $\mathcal{I}$ be a model of $\mathcal{T}$. Then $\mathcal{T}$ is* complete *w.r.t. $\mathcal{I}$ and M if the following are equivalent for all implications $A \to B$ over M:*

i) $A \to B$ holds in $\mathcal{I}$.
ii) $\sqcap A \sqsubseteq \sqcap B$ follows from $\mathcal{T}$.

Note that here we are interested only in the completeness of the TBox, unlike in [5], where the completeness of the TBox together with the ABox was considered. For a fixed set of interesting concepts, we say that the TBox is complete if it contains all relevant knowledge about implications between these concepts.

Definition 5. *For a given set of concept descriptions M, a model $\mathcal{I}$, and a set $A \subseteq M$, we say that A is* closed *under subsumption relationships over M if A is closed under the set of implications over M that hold in $\mathcal{I}$.*

3 Horn Approximations

For measuring the quality of an approximation, we use the notion of ε-Horn approximation from [7], which was initially proposed in [12]. It is based on the difference between the set of models of the implications detected so far and the set of models of the actual implications that hold in the application domain.

Algorithm 1 IsMember(A, *is_valid*($\cdot$), $\mathcal{T}$)

```
    Input: A set $A \subseteq M$, an implication oracle is_valid() for some $\mathcal{I}$, TBox $\mathcal{T}$ with model $\mathcal{I}$.
    Output: true if $A$ is closed under the subsumption relationships over $M$ that hold in $\mathcal{I}$, false otherwise.
 1: if $\sqcap A \sqsubseteq_{\mathcal{T}} \bot$ then
 2:     return false
 3: for all $c \in M \setminus A$ do
 4:     if $\sqcap A \sqsubseteq_{\mathcal{T}} c$ then
 5:         return false
 6: if $is\_valid(A \to \bot)$ then
 7:     return false
 8: for all $c \in M \setminus A$ do
 9:     if $is\_valid(A \to \{c\})$ then
10:         return false
11: return true
```

Definition 6. *Let M be a set of concept descriptions, $\mathcal{L}$ be a set of implications over M, and $\hat{\mathcal{L}}$ be the set of implications over M that hold in an interpretation $\mathcal{I}$. Then we say that $\mathcal{L}$ is an ε-*Horn approximation of $\hat{\mathcal{L}}$ *and an ε-*Horn approximation of $\mathcal{I}$ w.r.t. M *if*

$$\frac{|Mod(\mathcal{L}) \bigtriangleup Mod(\hat{\mathcal{L}})|}{2^{|M|}} \leq \varepsilon$$

where $Mod(\mathcal{L})$ stands for the set of models of $\mathcal{L}$ and $A \bigtriangleup B$ is the symmetric difference between sets A and B.

Our approach needs only membership queries. However, in the classical attribute exploration method, the queries asked to the expert are so-called implication queries, as mentioned in Section 1. In fact, a membership query can be simulated by a polynomial number of implication queries. Let $\hat{\mathcal{L}}$ be the set of implications over M that hold in the expert's model $\mathcal{I}$. It is easy to see that a set $A \subseteq M$ is a member of $\text{Mod}(\hat{\mathcal{L}})$ iff the implication $A \to \{c\}$ holds in $\mathcal{I}$ for no $c \in M \setminus A$ [3, 7].

Algorithm 2 implements this idea. Additionally, before making a query to the expert, it first checks whether the query already follows from the TBox, which would spare the expert answering this query.

Next we simulate equivalence queries by using a stochastic procedure. We sample a certain number of subsets of M and check whether any of them is a counterexample. More precisely, we check if it is a model of the already computed set of implications $\mathcal{L}$, but is not closed under the set of implications that hold in the expert's application domain $\mathcal{I}$, or vice versa. If one of these is the case, then we return this subset as counterexample. Otherwise, we say that $\mathcal{L}$ is approximately equivalent to the set of implications that hold in $\mathcal{I}$. This gives us the Algorithm 1 for approximately checking equivalence originating from [1]. The only modification is in the subprocedure IsMember, which, as explained

Algorithm 2 IsApproximatelyEquivalent($\mathcal{L}$, $is\_valid(\cdot)$, $\mathcal{T}$, ε, δ, i)

```
   Input: A set of implications $\mathcal{L}$ over $M$, an implication oracle $is\_valid(\cdot)$ for some
   $\mathcal{I}$, TBox $\mathcal{T}$ with model $\mathcal{I}$, $0 < \varepsilon \leq 1$, $0 < \delta \leq 1$, and $i \in \mathbb{N}$.
   Output: A counterexample to $\mathcal{L}$ if found; true otherwise.
1: for $i := 1$ to $\lceil \frac{1}{\varepsilon} \cdot (i + ln\frac{1}{\delta}) \rceil$ do
2:     generate $X \subseteq M$ uniformly at random
3:     if $(X \models \mathcal{L}) \neq$ (IsMember($X, is\_valid(\cdot), \mathcal{T}$)) then
4:         return $X$
5: return true
```

above, first checks whether the implication under consideration already follows from the TBox.

Algorithm 1 samples $\lceil \frac{1}{\varepsilon} \cdot (i + ln\frac{1}{\delta}) \rceil$ subsets of M to simulate the ith equivalence query asked by Algorithm 3 below, where $1-\delta$ is the pre-specified probability that Algorithm 3 computes an ε-Horn approximation of valid subsumption relationships over M. For each generated subset X, Algorithm 1 checks if X models the set $\mathcal{L}$ of implications so far computed in Algorithm 3. (In the context of exact learning, this set of implications is called the hypothesis). Additionally, it checks whether X is closed under the subsumption relationships that hold in $\mathcal{I}$. If the answers to these two tests are different, then X is a counterexample to $\mathcal{L}$. If none of the generated subsets is a counterexample, Algorithm 1 concludes that $\mathcal{L}$ is an ε-approximation of the implications that hold in $\mathcal{I}$.

We are now ready to formulate an algorithm that, with a given probability and within a given error bound, approximates the subsumption relationships that hold in the expert's model $\mathcal{I}$. Based on the exact learning algorithm in [2] and its PAC version in [7], it starts with the empty set of implications $\mathcal{L}$ and proceeds until a positive answer is obtained from an approximate equivalence query. If a negative counterexample X is received instead, the algorithm uses membership queries to find an implication $A \to B$ in $\mathcal{L}$ such that $A \not\subseteq X$ and $A \cap X$ is not closed under the subsumption relationships that hold in $\mathcal{I}$. If such an implication is found, the implication $A \to B$ is replaced by $A \cap X \to B$, which ensures that X is no longer a model of $\mathcal{L}$; otherwise, the implication $X \to \bot$ is added to $\mathcal{L}$. When a positive counterexample X is obtained from an approximate equivalence query, every implication $A \to B$ of which X is not a model is relaxed in a conservative way: given that X is a model of the target formula, $A \to B$ is replaced by $A \to B \cap X$ if $B \neq \bot$ and by $A \to X$ if $B = \bot$. When the algorithm terminates, the set of implications $\mathcal{L}$ is, with probability at least $1-\delta$, an ε-Horn approximation of the implications that hold in the expert's model $\mathcal{I}$.

The termination and correctness of Algorithm 3 easily follow from the results in [2, 7]. Essentially, the only difference from the algorithm in [7] is in the subprocedure IsApproximatelyEquivalent. It validates the generated counterexamples not only with the expert, but also with the TBox by calling the procedure IsMember with $\mathcal{T}$ as a parameter.

Algorithm 3 PAC-HornApproximation(M, *is_valid*$(\cdot)$, $\mathcal{T}$, ε, δ)

```
    Input: A set $M$ of concept descriptions, an implication oracle is_valid() for some
    interpretation $\mathcal{I}$, a TBox $\mathcal{T}$ with model $\mathcal{I}$, $0 \le \varepsilon \le 1$, and $0 \le \delta \le 1$.
    Output: A set of implications $\mathcal{L}$ that, with probability at least $1-\delta$, is an $\varepsilon$-
    approximation of subsumption relations over $M$ that hold in $\mathcal{I}$.
 1: $\mathcal{L} := []$
 2: $i := 1$
 3: while IsApproximatelyEquivalent($\mathcal{L}, is\_valid(\cdot), \mathcal{T}, \varepsilon, \delta, i$) returns a counterex-
    ample $X$ do
 4:     if $X \models \mathcal{L}$ then                                   ▷ negative counterexample
 5:         $found$ := false
 6:         for all $A \to B \in \mathcal{L}$ do
 7:             $C := A \cap X$
 8:             if $A \neq C$ and not IsMember($C$, is_valid(·), $\mathcal{T}$) then
 9:                 replace $A \to B$ in $\mathcal{L}$ with $C \to B$
10:                 $found$ := true
11:         if not $found$ then
12:             add $X \to \bot$ to the end of $\mathcal{L}$
13:     else                                                        ▷ positive counterexample
14:         for all $A \to B \in \mathcal{L}$ s.t. $X \not\models A \to B$ do
15:             if $B = \bot$ then
16:                 replace $A \to B$ with $A \to X$
17:             else
18:                 replace $A \to B$ with $A \to B \cap X$
19:     $i := i + 1$
20: return $\mathcal{L}$
```

4 Approximate Completion of TBoxes

Algorithm 3 computes the set of implications that approximate the expert's model $\mathcal{I}$. However, our goal is not only to compute this set of implications, but also to extend the TBox with the GCIs corresponding to these implications. If we do it in a naïve way, we might end up with an inconsistent TBox. The reason is that $\mathcal{L}$, being an approximation of $\mathcal{I}$, might contain implications that do not hold in $\mathcal{I}$.

In order to overcome this problem, we do the following modification to the algorithm: before adding an implication to $\mathcal{L}$ or replacing an implication in $\mathcal{L}$, we query the validity of the implications of the form $X \to \{c\}$ for $c \in M \setminus X$, where X is the premise of the new implication (see Algorithm 4). Those c for which the answer is positive together with the concept descriptions from X form the closure of X under the subsumption relations valid in $\mathcal{I}$. We set the conclusion of the new implication equal to this closure. With this modification, $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{L}$ always hold and our sampling procedure in Algorithm 1 returns only negative counterexamples. Therefore, the part of Algorithm 3 that deals with positive counterexamples is left out in the modified version presented in Algorithm 5.

Algorithm 4 CLOSE(A, *is_valid*$(\cdot)$)

```
Input: A set $A \subseteq M$ and an implication oracle is_valid(·) for an interpretation $\mathcal{I}$.
Output: The closure of $A$ under the subsumption relations over $M$ that hold in $\mathcal{I}$.
1: if is_valid($A \to \bot$) then
2:     return $\bot$
3: $B := A$
4: for all $c \in M \setminus A$ do
5:     if is_valid($A \to \{c\}$) then
6:         $B := B \cup \{c\}$
7: return $B$
```

The following result follows from Theorem 2 in [7] assuming that $\mathcal{T}$ is formulated in a DL where reasoning is in polynomial time.

Theorem 1. *Algorithm 5 runs in time polynomial in* $|M|$, $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$, *and the minimal size of the logically complete set of subsumption relationships over* M *that hold in* $\mathcal{I}$. *It outputs a TBox* $\mathcal{T}$ *that, with probability at least* $1 - \delta$, *is an* ε*-Horn approximation of* $\mathcal{I}$ *w.r.t.* M.

5 Conclusion and future work

We have provided an algorithm for approximately computing the subsumption relationships (over a fixed set of concept descriptions) that hold in the model of a domain expert via asking implication questions to this expert. The implications that are detected in this way to be missing from the TBox are added to the TBox until the TBox approximately represents the view of the domain expert. Our method is based on the exact learning algorithm by Angluin et al. [2] and a PAC learning variant of this algorithm [7].

As future work, we are going to implement this approach as a prototype and test its usefulness in real-world application domains. One idea might be to extend the PROTÉGÉ plugin ONTOCOMP that was presented in [22] for completing knowledge bases using the approach in [5].

In our approach, we do not take into account background knowledge that can be derived from our choice of concept descriptions, such as $\{C, \neg C\} \to \bot$ for a concept description C. It would be interesting to define the notion of approximation relative to explicitly or implicitly given background knowledge and to design an algorithm for computing such approximations.

The other direction of our future work will be employing PAC learning for approximate learning of DL ontologies. The exact learning version of this problem has already been studied in detail in [13] and relevant fragments of $\mathcal{EL}$ and DL-Lite have been identified that allow for learnability of ontologies with polynomial number of queries. Similar to this, we are going to investigate how these results can be leveraged if approximate learning is aimed instead of exact learning.

Algorithm 5 PAC-TBox-Completion(M, $is\_valid(\cdot)$, $\mathcal{T}$, ε, δ)

```
    Input: A set $M$ of concept descriptions, an implication oracle is_valid() for some
    interpretation $\mathcal{I}$, a TBox $\mathcal{T}$ with model $\mathcal{I}$, $0 \leq \varepsilon \leq 1$, and $0 \leq \delta \leq 1$.
    Output: A TBox $\mathcal{T}$ that, with probability at least $1-\delta$, is an $\varepsilon$-Horn approximation
    of $\mathcal{I}$.
 1: $\mathcal{L} := []$
 2: $i := 1$
 3: while IsApproximatelyEquivalent($\mathcal{L}$, $is\_valid(\cdot)$, $\mathcal{T}$, $\varepsilon$, $\delta$, $i$) returns a negative
    counterexample $X$ do
 4:     $found :=$ false
 5:     for all $A \to B \in \mathcal{L}$ do
 6:         $C := A \cap X$
 7:         if $A \neq C$ and not IsMember($C$, $is\_valid(\cdot)$, $\mathcal{T}$) then
 8:             $D :=$ Close($C, is\_valid(\cdot)$)
 9:             replace $A \to B$ in $\mathcal{L}$ with $C \to D$
10:             replace $\sqcap A \sqsubseteq \sqcap B$ in $\mathcal{T}$ with $\sqcap C \sqsubseteq \sqcap D$
11:             $found :=$ true
12:     if not $found$ then
13:         $Y :=$ Close($X, is\_valid(\cdot)$)
14:         add $X \to Y$ to the end $\mathcal{L}$
15:         add $\sqcap X \sqsubseteq \sqcap Y$ to $\mathcal{T}$
16:     $i := i + 1$
17: return $\mathcal{T}$
```

Acknowledgements

Sergei Obiedkov is supported by the Russian Science Foundation (grant 17-11-01294).

References

1. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
2. D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9:147–164, 1992.
3. M. Arias, J. L. Balcázar, and C. Tirnăucă. Learning definite horn formulas from closure queries. *Theor. Comput. Sci.*, 658:346–356, 2017.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
5. F. Baader, B. Ganter, B. Sertkaya, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. In M. M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 230–235. AAAI Press, 2007.
6. F. Baader, I. Horrocks, C. Lutz, and U. Sattler, editors. *An Introduction to Description Logic.* Cambridge University Press, 2017.
7. D. Borchman, T. Hanika, and S. Obiedkov. Probably approximately correct learning of horn envelopes from queries. *Discrete Applied Mathematics*, 2019. To appear.

8. B. Ganter. Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany, 1984.
9. B. Ganter and S. Obiedkov. *Conceptual Exploration*. Springer, Berlin/Heidelberg, 2016.
10. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, Germany, 1999.
11. A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in OWL ontologies. In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications. Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2006.
12. H. A. Kautz, M. J. Kearns, and B. Selman. Horn approximations of empirical data. *Artificial Intelligence*, 74(1):129–145, 1995.
13. B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. In C. Baral, G. De Giacomo, and T. Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria*. AAAI Press, 2014.
14. B. Konev, C. Lutz, A. Ozaki, and F. Wolter. Exact learning of lightweight description logic ontologies. *Journal of Machine Learning Research*, 18:201:1–201:63, 2017.
15. S. Obiedkov. Learning implications from data and from queries. In D. Cristea, F. L. Ber, and B. Sertkaya, editors, *Proceedings of the 15th International Conference on Formal Concept Analysis, (ICFCA 2019)*, Lecture Notes in Artificial Intelligence, 2019. To appear.
16. R. Peñaloza and B. Sertkaya. Axiom pinpointing is hard. In B. C. Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Proceedings of the 2009 International Workshop on Description Logics (DL2009)*, volume 477 of *CEUR-WS*, 2009.
17. R. Peñaloza and B. Sertkaya. Complexity of axiom pinpointing in the DL-Lite family of description logics. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 29–34. IOS Press, 2010.
18. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in the $\mathcal{EL}$ family of Description Logics. In F. Lin, U. Sattler, and M. Truszczynski, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, (KR 2010)*, pages 280,289. AAAI Press, 2010.
19. R. Peñaloza and B. Sertkaya. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artif. Intell.*, 250:80–104, 2017.
20. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 355–362. Morgan Kaufmann, 2003.
21. S. Schlobach, Z. Huang, R. Cornet, and F. Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007.
22. B. Sertkaya. Ontocomp: A protege plugin for completing owl ontologies. In L. Aroyo and P. Traverso, editors, *Proceedings of the 6th European Semantic Web Conference, (ESWC 2009)*, volume 5554 of *Lecture Notes in Computer Science*, pages 898–902. Springer-Verlag, 2009.