Learning Probabilistic Description Logics: A Framework and Algorithms

José Eduardo Ochoa Luna[1], Kate Revoredo[2], and Fabio Gagliardi Cozman[1]

[1] Escola Politécnica, Universidade de São Paulo,
Av. Prof. Mello Morais 2231, São Paulo - SP, Brazil
[2] Departamento de Informática Aplicada, Unirio
Av. Pasteur, 458, Rio de Janeiro, RJ, Brazil
eduardo.ol@gmail.com,katerevoredo@uniriotec.br,fgcozman@usp.br

Abstract. Description logics have become a prominent paradigm in knowledge representation (particularly for the Semantic Web), but they typically do not include explicit representation of uncertainty. In this paper, we propose a framework for automatically learning a Probabilistic Description Logic from data. We argue that one must learn both concept definitions and probabilistic assignments. We also propose algorithms that do so and evaluate these algorithms on real data.

1 Introduction

Description logics (DLs) [2] form a family of knowledge representation formalisms that model the application domain by defining the relevant concepts of the domain and then using these concepts to specify properties of objects and relation among concepts. Even though DLs are quite expressive, they have limitations, particularly when it comes to modeling uncertainty. Thus probabilistic extensions to DLs have been proposed, defining different Probabilistic Description Logics (PDLs). For instance, the PDL CR$\mathcal{ALC}$ [6, 22, 7] allows one to perform probabilistic reasoning by adding uncertainty capabilities to the DL $\mathcal{ALC}$ [2].

PDLs have been extensively investigated in the last few years [5, 8, 19]. To build a PDL terminology with large amounts of data, one must invest considerable resources. Thus, machine learning algorithms can be used in order to automatically learn a PDL. To the best of our knowledge, the only proposals for learning PDLs were described in [20] and [24]. Both focused on learning CR$\mathcal{ALC}$, but the former focused on learning concept definitions and the latter on probabilistic inclusions.

In this paper, we argue that to completely learn a PDL one must learn its concept definitions *and* probabilistic inclusions. We expect that learning algorithms can accommodate together background knowledge and deterministic and probabilistic concepts, giving each component its due relevance. Therefore, we propose a framework for automatically learning a PDL from relational data. Focusing on CR$\mathcal{ALC}$, we propose algorithms that do so and evaluate these algorithms on real data; compared to the existing work mentioned in the previous paragraph, we

This is a preprint version of a paper published in:
Lecture Notes in Artificial Intelligence, vol. 7094, Springer, 2011.
When citing, please cite the final version published by Springer.

contribute by presenting a more complete framework, and by comparing several algorithms in our experiments.

The paper is organized as follows. Section 2 reviews basic concepts of DLs, PDLs, CR$\mathcal{ALC}$ and machine learning in a deterministic setting. Section 3 presents our algorithm for PDL learning. Experiments are discussed in Section 4, and Section 5 concludes the paper.

2 Basics

In this section we briefly review both deterministic and probabilistic components of PDL.

2.1 Description Logics

Description logics (DLs) form a family of representation languages that are typically decidable fragments of first order logic (FOL) [2]. Knowledge is expressed in terms of *individuals*, *concepts*, and *roles*. The semantics of a description is given by a *domain* $\mathcal{D}$ (a set) and an *interpretation* $\cdot^{\mathcal{I}}$ (a functor). Individuals represent objects through names from a set $N_{\mathsf{I}} = \{a, b, \ldots\}$. Each *concept* in the set $N_{\mathsf{C}} = \{C, D, \ldots\}$ is interpreted as a subset of a domain $\mathcal{D}$. Each *role* in the set $N_{\mathsf{R}} = \{r, s, \ldots\}$ is interpreted as a binary relation on the domain.

Concepts and roles are combined to form new concepts using a set of *constructors*. Constructors in the $\mathcal{ALC}$ logic are *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$), and *value* restriction ($\forall r.C$). *Concept inclusions/definitions* are denoted respectively by $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. Concepts ($C \sqcup \neg C$) and ($C \sqcap \neg C$) are denoted by $\top$ and $\bot$ respectivelly. Information is stored in a *knowledge base* ($\mathcal{K}$) divided in two parts: the TBox (terminology) and the ABox (assertions). The TBox lists concepts and roles and their relationships. A TBox is acyclic if it is a set of concept inclusions/definitions such that no concept in the terminology uses itself. The ABox contains assertions about objects.

Given a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$, the reasoning services typically include (i) consistency problem (to check whether the $\mathcal{A}$ is consistent with respect to the $\mathcal{T}$); (ii) entailment problem (to check whether an assertion is entailed by $\mathcal{K}$; note that this generates class-membership assertions $\mathcal{K} \models C(a)$, where a is an individual and C is a concept); (iii) concept satisfiability problem (to check whether a concept is subsumed by another concept with respect to the $\mathcal{T}$). The latter two reasoning services can be reduced to the consistency problem [2].

2.2 Probabilistic Description Logics and CR$\mathcal{ALC}$

Several probabilistic descriptions logics (PDLs) have appeared in the literature. Heinsohn [11], Jaeger [13] and Sebastiani [25] consider probabilistic inclusion axioms such as $P_{\mathcal{D}}(\mathsf{Professor}) = \alpha$, meaning that a randomly selected object is a

Professor with probability α. This characterizes a *domain-based* semantics: probabilities are assigned to subsets of the domain $\mathcal{D}$. Sebastiani also allows inclusions such as $P(\mathsf{Professor}(\mathsf{John})) = \alpha$, specifying probabilities over the interpretations themselves. For example, one interprets $P(\mathsf{Professor}(\mathsf{John})) = 0.001$ as assigning 0.001 to be the probability of the set of interpretations where John is a Professor. This characterizes an *interpretation-based* semantics.

The PDL CR$\mathcal{ALC}$ is a probabilistic extension of the DL $\mathcal{ALC}$ that adopts an interpretation-based semantics. It keeps all constructors of $\mathcal{ALC}$, but only allows concept names on the left hand side of inclusions/definitions. Additionally, in CR$\mathcal{ALC}$ one can have probabilistic inclusions such as $P(C|D) = \alpha$ or $P(r) = \beta$ for concepts C and D, and for role r. If the interpretation of D is the whole domain, then we simply write $P(C) = \alpha$. The semantics of these inclusions is roughly (a formal definition can be found in [7]) given by:

$$\forall x \in \mathcal{D} \ : \ P(C(x)|D(x)) = \alpha,$$

$$\forall x \in \mathcal{D}, y \in \mathcal{D} \ : \ P(r(x,y)) = \beta.$$

We assume that every terminology is acyclic; no concept uses itself. This assumption allows one to represent any terminology $\mathcal{T}$ through a directed acyclic graph. Such a graph, denoted by $\mathcal{G}(\mathcal{T})$, has each concept name and role name as a node, and if a concept C directly uses concept D, that is if C and D appear respectively in the left and right hand sides of an inclusion/definition, then D is a *parent* of C in $\mathcal{G}(\mathcal{T})$. Each existential restriction $\exists r.C$ and value restriction $\forall r.C$ is added to the graph $\mathcal{G}(\mathcal{T})$ as nodes, with an edge from r to each restriction directly using it. Each restriction node is a *deterministic* node in that its value is completely determined by its parents.

The semantics of CR$\mathcal{ALC}$ is based on probability measures over the space of interpretations, for a fixed domain. Inferences, such as $P(\mathsf{A}_o(\mathsf{a}_0)|\mathcal{A})$ for an ABox $\mathcal{A}$, can be computed by propositionalization and probabilistic inference (for exact calculations) or by a first order loopy propagation algorithm (for approximate calculations) [7].

2.3 Learning Description Logics

The use of ontologies for knowledge representation has been a key element of proposals for the Semantic Web [1, 9]. Considerable effort is currently invested into developing automated means for the acquisition of ontologies [16].

Most early approaches were only capable of learning simple ontologies such as taxonomic hierarchies. Some recent approaches such as YINYANG [12], DL-FOIL [9] and DL-Learner [18] have focused on learning expressive terminologies (we refer to [20] for a detailed review on learning description logics). To some extent, all these approaches have been inspired by Inductive Logic Programming (ILP) [15] techniques, in that they try to transfer ILP methods to description logic settings. The goal of learning in such deterministic languages is generally to find a correct concept with respect to given examples. Given a knowledge base $\mathcal{K}$, a target concept Target such that $\mathsf{Target} \notin \mathcal{K}$, a set $E = E_p \cup E_n$ of positive

and negative examples given as assertions for Target, the goal of learning is to find a concept definition $C(\mathsf{Target} \equiv C)$ such that $\mathcal{K} \cup C \models E_p$ and $\mathcal{K} \cup C \not\models E_n$.

A sound concept definition for Target must cover all positive examples and none of the negative examples. A learning algorithm can be constructed as a combination of (1) a refinement operator, which defines how a search tree can be built, (2) a search algorithm, which controls how the tree is traversed, and (3) a scoring function to evaluate the nodes in the tree defining the best one.

Refinement operators allow one to find candidate concept definitions through two basic tasks: generalization and specialization [17]. Such operators in both ILP and description logic learning rely on θ-subsumption to establish an ordering so as to traverse the search space. If a concept C subsumes a concept D ($D \sqsubseteq C$), then C covers all examples which are covered by D, which makes subsumption a suitable order. Arguably the best refinement operator for description logic learning is the one available in the DL-Learner system [17, 18].

In a deterministic setting a cover relationship simply tests whether, for given candidate concept definition (C), a given example e holds; that is, $\mathcal{K} \cup C \models e$ where $e \in E_p$ or $e \in E_n$. In this sense, a cover relationship $cover(e, \mathcal{K}, C)$ indicates whether a candidate concept covers a given example [9].

In DL learning, one often compares candidates through score functions based on the number of positive/negative examples covered. In DL-Learner a fitness relationship considers the number of positive examples as well as the length of solutions when expanding candidates in the tree search.

The learning algorithm depends basically on the way we traverse the candidate concepts obtained after applying refinement operators. In a deterministic setting the search for candidate concepts is often based on the FOIL [23] algorithm. There are also different approaches (for instance, DL-Learner, an approach based on genetic algorithms [16], and one that relies on horizontal expansion and redundance checking to traverse search trees [18]).

3 Learning with the PDL CR$\mathcal{ALC}$

A probabilistic terminology consists of both concepts definitions and probabilistic components (probabilistic inclusions in CR$\mathcal{ALC}$). The key in learning under a combined approach is to give a due relevance to each component. In this section we combine existing algorithms into a single framework that allows for logical and probabilistic learning.

We argue that negative and positive examples underlie the choice of either a concept definition or a probabilistic inclusion. In a deterministic setting we expect to find concepts covering all positive examples, which is not always possible. It is common to allow flexible heuristics that deal with these issues. Moreover, there are several examples that cannot be ascribed to candidate hypotheses[3].

[3] In some cases the Open World Assumption inherent to description logics prevent us for stating membership of concepts.

When we are unable to find a concept definition that covers all positive examples we assume such hypothesis as candidates to be a probabilistic inclusion and we begin the search for the best probabilistic inclusion that fits the examples.

As in description logic learning three tasks are important and should be considered: (1) refinement operators, (2) scoring functions and (3) a traverse search space algorithm.

The refinement operator described previously is used for learning the deterministic component of probabilistic terminologies.

3.1 The Probabilistic Score Function

In our proposal, since we want to learn probabilistic terminologies, we adopt a probabilistic cover relation given in [14]:

$$cover(e, \mathcal{K}, C) = P(e|\mathcal{K}, C).$$

Every candidate hypothesis together with a given example turns out to be a probabilistic random variable which yields true if the example is covered, and false otherwise. To guarantee soundness of the ILP process (that is, to cover positive examples and not to cover negative examples), the following restrictions are needed:

$$P(e_p|\mathcal{K}, C) > 0, \qquad P(e_n|\mathcal{K}, C) = 0.$$

In this way a probabilistic cover relationship is a generalization of the deterministic cover, and is suitable for a combined approach. Probabilities can be computed through Bayes' theorem:

$$P(e|\mathcal{K}, C_1, \ldots, C_k) = \frac{P(C_1, C_2, \ldots, C_k|T)P(T)}{P(C_1, \ldots, C_k)},$$

where $C_1, \ldots, C_k$ are candidate concepts definitions, and T denotes the target concept variable. Here are three possibilities for modeling $P(C_1, \ldots, C_k|T)$: (1) a naive Bayes assumption may be adopted [14] (each candidate concept is independent given the target), and then $P(C_1, \ldots, C_k|T) = \prod_i P(C_i|T)$; (2) the noisy-OR function may be used [20]; (3) a less restrictive option based on tree augmented naive Bayes networks (TAN) may be handy [14]. This last possibility has been considered for the probabilistic cover relationship used in this paper. In each case probabilities are estimated by maximum (conditional) likelihood parameters. The candidate concept definition C_i with the highest probability $P(C_i|T)$ is the one chosen as the best candidate.

As we have chosen a probabilistic cover relationship, our probabilistic score is defined accordingly:

$$score(\mathcal{K}|C) = \prod_{e_i \in E_p} P(e_i|\mathcal{K}, C),$$

where C is the best candidate chosen as described before.

In the probabilistic score we have previously defined, a given threshold allows us to differentiate between a deterministic and probabilistic inclusion candidate. Further details are given in the next section.

3.2 The Algorithm to Learn Probabilistic Terminologies

The choice between a deterministic or a probabilistic inclusion is based on a probabilistic score. We start by searching a deterministic concept. If after a set of iterations the score of the best candidate is below a given threshold, searching for a probabilistic inclusion is better than to keep searching for a deterministic concept definition. Then, the current best k-candidates are considered as start point for probabilistic inclusion search. The complete learning procedure is shown in Algorithm 1.

Require: an initial knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$, a target concept $C_{\mathcal{T}}$ and a training set E.
1: SearchTree with a node $\{C = \top, h = 0\}$
2: **repeat**
3: choose node $N = \{C, h\}$ with highest probabilistic score in SearchTree
4: expand node to length $h + 1$:
5: add all nodes $D \in$ (refinementOperator(C)) with length $= h + 1$
6: learn parameters for all nodes D
7: $N = \{C, h + 1\}$
8: expand alternative nodes according to horizontal expansion factor and $h + 1$[18]
9: **until** stopping criterion
10: $N' =$ best node in SearchTree
11: **if** score(N') > threshold **then**
12: return deterministic concept $C' \in N'$ ($C_{\mathcal{T}} \equiv C'$)
13: **else**
14: call ProbabilisticInclusion(SearchTree, $C_{\mathcal{T}}$)
15: **end if**

Algorithm 1: Algorithm for learning probabilistic terminologies.

The algorithm starts with an overly general concept definition in the root of the search tree (line 1). This node is expanded according to refinement operators and horizontal expansion criterion (line 4), i.e, child nodes obtained by refinement operators are added to the search tree (line 5). The probabilistic parameters of these child nodes are learned (line 6) and then they are evaluated with the best one chosen for a new expansion (line 3) (alternative nodes based on horizontal expansion factor are also considered (line 8)). This process continues until a stopping criterion is attained (difference for scores is insignificant); After that, the best node obtained is evaluated and if it is above a threshold, a deterministic concept definition is found and returned (line 12). Otherwise, a probabilistic inclusion procedure is called (line 14).

The Algorithm 2 learns probabilistic inclusions. It starts retrieving the best k nodes in the search tree and computing the conditional mutual information for every pair of nodes (line 2). Then an undirected graph is built where the vertices are the k nodes and the edges are weighted with the value of the conditional mutual information [21] for each pair of vertices (lines 4 and 5). A maximum weight spanning tree [4] from this graph is built (line 6) and the target concept

is added as a parent for each vertice (line 7). The probabilistic parameters are learned (line 8). This learned TAN-based classifier [10] is used to evaluate the possible probabilistic inclusion candidates (line 9) and the best one is returned.

Require: SearchTree previously computed and a target concept $C_{\mathcal{T}}$
1: **for** each pair of candidates C_i, C_j in first k nodes of the SearchTree **do**
2: compute the conditional mutual information $I(C_i, C_j|C_{\mathcal{T}})$
3: **end for**
4: build an undirected graph in which vertices are the k candidates
5: annotate the weight of an edge connecting C_i to C_j by the $I(C_i, C_j|C_{\mathcal{T}})$
6: build a maximum weight spanning tree from this graph
7: add $C_{\mathcal{T}}$ as parent for each C_i
8: learn probabilities for $P(C_i|Parents(C_i))$
9: return the highest probabilistic inclusion $P(C_{\mathcal{T}}|C') = \alpha$

Algorithm 2: Algorithm for learning probabilistic inclusions.

4 Experiments

In order to evaluate the proposed framework we have divided the analysis in two stages. In a first stage, the framework was compared with the arguably best description logic learning algorithm available (the DL-Learner system). The second stage evaluated suitability of the framework for learning probabilistic CR$\mathcal{ALC}$ terminologies in real world domains. In this sense, comparisons with two previous approaches [20, 24] were performed. Experiments were run on a CORE 2 DUO 2.2 GHz computer with 4GB memory over a Linux Ubuntu platform. Each stage is detailed as follows.

4.1 Experiments on Description Logic Learning

The aim of the first stage was to investigate whether by introducing a probabilistic setting the framework behaves as well as traditional deterministic approaches in description logic learning tasks. In this preliminar evaluation (generally speaking, there is a lack of evaluation standards in ontology learning [18]) we have considered five datasets available in the DL-Learner system and reported in [18]. After a five-fold cross validation process, both accuracy and definitions length were measured. Evaluation results are shown in Table 1.

The best solution is the one that has both highest accuracy and shortest length. In Table 1, three algorithms are compared: (DL-Learner) the best description logic learning algorithm; (Noisy) a probabilistic Noisy-OR approach for learning CR$\mathcal{ALC}$ [20] with focus on concept learning and (Framework) the combined approach proposed in this paper. The Noisy algorithm had a slightly lower performance than DL-Learner in terms of accuracy. On the other hand, the Framework (combined approach) obtained similar accuracies like DL-Learner.

Table 1. Learning description logics results. Accur. (Accuracy) and Length (solution length)

	DL-Learner		Noisy		Framework	
Problem	Accur.	Length	Accur.	Length	Accur.	Length
trains	100	5	99.5	7	100	6
moral I	97.8	3	96.5	5	97.7	4
moral II	97.8	8	96.0	8	97.6	8
poker I	100	5	98.8	6	100	5
poker II	100	11	99.7	12	100	11

Further differences are observed when solution lengths are under analysis. The DL-Learner algorithm is concerned with finding shorter solutions (longer solutions are penalized in DL-Learner) whereas in the Noisy algorithm this issue is neglected. Conversely, the Framework aims at finding suitable shorter solutions because it also uses such a penalization scheme.

Despite of the fact that the framework is focused on learning probabilistic terminologies it is worth noting that two of the tested algorithms (mainly the Framework) can be competitive for learning description logics. It should be noted, however, that some induced definitions can be only meaningful when a probabilistic component is attached.

4.2 Experiments on Learning Probabilistic Terminologies

In this second stage we focused on learning of CR$\mathcal{ALC}$ probabilistic terminologies from real world data. To do so two public data sources were explored: the Lattes curriculum platform and the YAGO-Wikipedia ontology. Due to the lack of previous learning results, comparisons have been performed by measuring accuracy and fitness of probabilistic terminologies to data. In addition, two previous approaches for learning CR$\mathcal{ALC}$ [20, 24] have also been evaluated.

In the first set of experiments, datasets were constructed from information extracted of the Lattes curriculum platform [4]. This repository offers public relational data from Brazilian researchers. Information is given in HTML format so a parsing procedure was run to extract assertions on concepts and roles. Examples include name, institution, languages, address, etc. It was focused extraction of relational information through publication analysis, works supervised, examination board participations and so on. The universe was restricted to 1050 researchers. In Table 2 some features of the Lattes datasets are given.

After a 10-fold cross validation process, accuracy (Accur.) and amount of nodes originated (# Nodes) when constructing relational Bayesian network were considered for comparisons. Table 3 shows some experimental results obtained in these Lattes datasets. The best solution is the one that returns the highest accuracy and the fewest amount of nodes. The Noisy algorithm [20] got the

[4] http://lattes.cnpq.br

Table 2. Lattes curriculum datasets.

Problem	axioms	concepts	roles	examples
lattes I	2494	14	12	208
lattes II	2603	14	10	146
lattes III	2650	13	10	230
lattes IV	2350	2	13	34

worst values on accuracy and fewer amount of possible nodes. The Inclusion algorithm (which focuses learning of probabilistic inclusions [24]) got better results in accuracy than Noisy, however, it returned a greater amount of nodes. Finally, the Framework proposed (combined algorithm) obtained the best results on accuracy and a lower amount of nodes like the Noisy algorithm.

Table 3. Results of CR$\mathcal{ALC}$ learning (Lattes). Accur. (accuracy) and # Nodes (amount of nodes of relational Bayesian network).

	Noisy		Inclusion		Framework	
Problem	Accur.	# Nodes	Accur.	# Nodes	Accur.	# Nodes
lattes I	76.74	48	76.84	51	77.13	49
lattes II	69.52	54	72.77	58	74.42	56
lattes III	73.48	56	74.68	62	78.45	58
lattes IV	71.5	78	71.56	110	72.34	75

Further experiments were performed on datasets obtained from Wikipedia-YAGO ontology. Wikipedia articles consist mostly of free text, but also contain various types of structured information in the form of Wiki markup. Such information includes infobox templates, categorization information, images geocoordinates, links to external Web pages, disambiguation pages, redirects between pages, and links across different language editions of Wikipedia.

In the last years, there were several projects aimed at structuring such huge source of knowledge. Examples include The DBpedia project [3], which extracts structured information from Wikipedia and turns it into a rich knowledge base, and YAGO [26], a semantic knowledge base based on data from Wikipedia and WordNet[5]. Currently, YAGO knows more than 2 million entities (like persons, organizations, cities, etc.). It knows 20 million facts about these entities. Unlike many other automatically assembled knowledge bases, YAGO has a manually confirmed accuracy of 95%. Several domains ranging from films, places, historical events, wines, etc. have been considered in this ontology. Moreover, facts are given as binary relationships that are suitable for our learning settings.

[5] wordnet.princeton.edu/

Subsets of Wikipedia-YAGO facts were used for learning probabilistic CR$\mathcal{ALC}$ terminologies. Two domains were focused: first, about scientists (1335); second, about film directors (2000). Table 4 shows some features of these datasets.

Table 4. Dataset Wikipedia-YAGO.

Problem	axioms	concepts	roles	examples
scientists I	7180	11	32	438
scientists II	6534	11	32	378
directors I	8006	12	35	626
directors II	7234	12	35	389

In Table 5 results obtained with these datasets are exhibited. After a 10-fold cross validation process, accuracy and amount of nodes when creating relational Bayesian network were evaluated. Learning in these datasets was a more complex task than Lattes. Indeed, it can be inferred of the accuracy values (lower values) and the amount of nodes obtained. Meanwhile, the same patterns were observed, i.e., the Framework returned the best results in accuracy and a lower amount of nodes. On the other hand, the Inclusion algorithm had better accuracy than Noisy but a greater amount of nodes.

Table 5. Results of CR$\mathcal{ALC}$ learning (Wikipedia-YAGO). Accur. (accuracy) and # Nodes (amount of nodes in relational Bayesian network).

	Noisy		Inclusion		Framework	
Problem	Accur.	# Nodes	Accur.	# Nodes	Accur.	# Nodes
scientists I	81.4	53	82.33	56	83.44	55
scientists II	82.3	65	82.46	69	84.55	68
directors I	65.67	67	67.14	69	73.56	66
directors II	69.45	74	71.3	79	74.75	76

By evaluating these two real world datasets, we conclude that the combined framework outerperforms the other approaches in learning probabilistic CR$\mathcal{ALC}$ terminologies. Our proposal obtains balanced results in accuracy and structural complexity.

5 Conclusion

We have presented a framework for learning deterministic/probabilistic components of terminologies expressed in CR$\mathcal{ALC}$, and contributed with experiments that show the value of the framework. This unified learning scheme employs a

refinement operator for traversing the search space, a probabilistic cover and score relationships for evaluating candidates, and a mixed search procedure. The search aims at finding deterministic concepts; if the score obtained is below a given threshold, a probabilistic inclusion search is conducted (a probabilistic classifier is produced). Experiments with probabilistic terminologies in two real-world domains suggest that this framework do lead to improved likelihoods.

We note that the current literature does not explore in depth the use of data to learn knowledge bases in description logics that can handle uncertainty explicitly. The present contribution is a step in this direction.

Acknowledgements

The third author is partially supported by CNPq. The work reported here has received substantial support through FAPESP grant 2008/03995-5.

References

1. G. Antoniou and F. van Harmelen. *Semantic Web Primer*. MIT Press, 2008.
2. F. Baader and W. Nutt. Basic description logics. In *Description Logic Handbook*, pages 47–100. Cambridge University Press, 2002.
3. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, 2009.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
5. P. C. G. Costa and K. B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Proceeding of the 2006 conference on Formal Ontology in Information Systems*, pages 237–249, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.
6. F. G. Cozman and R. B. Polastro. Loopy propagation in a probabilistic description logic. In Sergio Greco and Thomas Lukasiewicz, editors, *Second International Conference on Scalable Uncertainty Management*, Lecture Notes in Artificial Intelligence (LNAI 5291), pages 120–133. Springer, 2008.
7. F. G. Cozman and R. B. Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Conference on Uncertainty in Artificial Intelligence*, 2009.
8. C. D'Amato, N. Fanizzi, and T. Lukasiewicz. Tractable reasoning with Bayesian description logics. In *SUM '08: Proceedings of the 2nd international conference on Scalable Uncertainty Management*, pages 146–159, Berlin, Heidelberg, 2008. Springer-Verlag.
9. N. Fanizzi, C. D'Amato, and F. Esposito. DL-FOIL concept learning in description logics. In *ILP '08: Proceedings of the 18th International Conference on Inductive Logic Programming*, pages 107–121, Berlin, Heidelberg, 2008. Springer-Verlag.
10. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
11. J. Heinsohn. Probabilistic description logics. In *International Conf. on Uncertainty in Artificial Intelligence*, pages 311–318, 1994.

12. L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.
13. M. Jaeger. Probabilistic reasoning in terminological logics. In *Principals of Knowledge Representation (KR)*, pages 461–472, 1994.
14. N. Landwehr, K. Kersting, and L. DeRaedt. Integrating Naïve Bayes and FOIL. *J. Mach. Learn. Res.*, 8:481–507, 2007.
15. N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.
16. J. Lehmann. Hybrid learning of ontology classes. In *Proceedings of the 5th International Conference on Machine Learning and Data Mining*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.
17. J. Lehmann and P. Hitzler. Foundations of refinement operators for description logics. In Hendrick Blockeel, Jude W. Shavlik, and Prasad Tadepalli, editors, *ILP '07: Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2007.
18. J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In Hendrick Blockeel, Jude W. Shavlik, and Prasad Tadepalli, editors, *ILP '07: Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2007.
19. T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6-7):852–883, 2008.
20. J. Ochoa-Luna and F.G. Cozman. An algorithm for learning with probabilistic description logics. In *Bobillo, F., et al. (eds.) Proceedings of the 5th International Workshop on Uncertainty Reasoning for the Semantic Web*, volume 527, pages 63–74, Chantilly, USA, 2009. CEUR-WS.org.
21. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference*. Morgan Kaufman, 1988.
22. R. B. Polastro and F. G. Cozman. Inference in probabilistic ontologies with attributive concept descriptions and nominals. In *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 7th International Semantic Web Conference (ISWC)*, Karlsruhe, Germany, 2008.
23. J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20. Springer-Verlag, 1993.
24. K. Revoredo, J. Ochoa-Luna, and F. Cozman. Learning terminologies in probabilistic description logics. In Antônio da Rocha Costa, Rosa Vicari, and Flavio Tonidandel, editors, *Advances in Artificial Intelligence SBIA 2010*, volume 6404 of *Lecture Notes in Computer Science*, pages 41–50. Springer / Heidelberg, Berlin, 2010.
25. F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 122–130, 1994.
26. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.