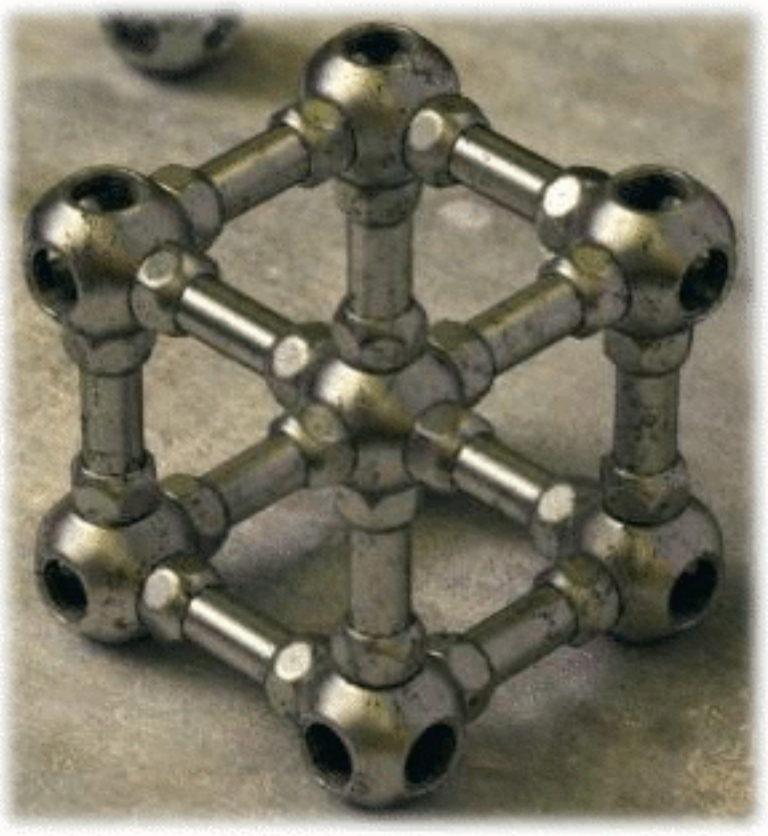

Learning Terminologies in Probabilistic Description Logics [1]

Author(s):

Kate Revoredo

José Eduardo Ochoa-Luna

Fabio Gagliardi Cozman

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.

Learning Terminologies in Probabilistic Description Logics

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

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

Abstract. This paper investigates learning methods where the target language is the recently proposed probabilistic description logic CR$\mathcal{ALC}$. We start with an inductive logic programming algorithm that learns logical constructs; we then develop an algorithm that learns probabilistic constructs by searching for conditioning concepts, using examples given as interpretations. Issues on learning from entailments are also examined, and practical examples are discussed.

1 Introduction

A description logic offers a formal language where one can describe concepts such as "A Professor is a Person who teaches in a University". To do so, a description logic typically uses a decidable fragment of first-order logic, trying to reach a practical balance between expressivity and complexity. The last decade has seen a significant increase in interest in description logics as a vehicle for large-scale knowledge representation, for instance in the semantic web. Indeed, the language OWL [1], proposed by the W3 consortium as the data layer of their architecture for the semantic web, is an XML encoding for an expressive description logics.

Description logics are not geared towards the representation of uncertainty about objects and concepts: one cannot express that "with low probability a Person is a Professor". The literature contains a number of proposals that add probabilistic uncertainty to description logics, as this is central to the management of semantic data in large repositories. Cozman and Polastro have proposed [3] a probabilistic extension of the popular logic $\mathcal{ALC}$ [2], called Credal $\mathcal{ALC}$ (CR$\mathcal{ALC}$), where sentences such as $P(\mathsf{Professor}|\mathsf{Researcher}) = 0.4$, refereing to the probability that an object is a Professor given that it is a Researcher, are allowed. These sentences are called *probabilistic inclusions*. Inference algorithms based on Relational Bayesian networks [12] have been proposed; because exact inference does not seem to be scalable when quantified concepts are employed, approximate inference algorithms have been developed [3,4].

An important question is how to automatically learn concepts expressed with description logics. Previous efforts with concept learning in CR$\mathcal{ALC}$ [14] have

A.C. da Rocha Costa, R.M. Vicari, F. Tonidandel (Eds.): SBIA 2010, LNAI 6404, pp. 41–50, 2010.

focused only on logical concept learning, using the ideas of refinement operators from DL-Learner [13] and DL-FOIL [9], and not dealing with probabilistic inclusions. In this paper, we assume that (logical) concept definitions can be learned using any algorithm in the literature and we attack the problem of learning probabilistic inclusions. For that, we propose an algorithm that learns the structure of the probabilistic inclusion by discovering how a concept is probabilistically conditioned on other concepts and then by estimating the probability of the probabilistic inclusion. The proposed algorithm learns from examples given as interpretations. We also discuss the generalization of these ideas to other settings; namely, learning from entailments and learning from proofs [7].

Section 2 provides background knowledge on description logics (Section 2.1), on probabilistic description logics (Section 2.2) and on relevant learning algorithms based on Inductive Logic Programming (Section 2.3). Section 3 proposes a learning algorithm for CR$\mathcal{ALC}$ probabilistic inclusions and Section 4 reports on some preliminaries experiments. Section 5 concludes the paper.

2 Background

Assume we are given a repository of HTML pages where researchers and students have stored data about publications, courses, languages. We might wish to extract the definition of some concepts, such as Researcher and Person, and relationships among them. Suppose however that we are unable to state deterministic relations among concepts, but instead we can only give probabilistic inclusions such as $P(\mathsf{Professor}|\mathsf{Researcher}) = 0.4$. Probabilistic inclusions are allowed in probabilistic description logics, as reviewed in this section.

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* Δ (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 [10], Jaeger [11] and Sebastiani [16] 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 $P(\mathsf{Professor}(\mathsf{John})) = \alpha$ as well, 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 all 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 in 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. For any element of the domain, the probability that this element is in C, given that it is in D is α. 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 [4]) 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) [4].

2.3 Learning in Description Logics

The techniques developed in this paper are inspired by two systems based on inductive logic programming (ILP) [5], namely, DL-Learner [13] and DL-FOIL [9]. The goal in ILP is to find an hypothesis H (a definite clause program) that covers all examples in a set E_p of positive examples and none examples in a set E_n of negative examples. In the context of DLs, the hypothesis is a knowledge base and the examples are assertions. Consider a concept name Target, a knowledge base $\mathcal{K}$ not containing Target, and sets of positive and negative examples with elements of the form $\mathsf{Target}(a)$, where a is an object of the domain $\mathcal{D}$. Acyclicity is assumed; that is, recursive inclusions/definitions are not learned. The learning problem is to find a concept C such that Target does not occur in C and for $\mathcal{K}' = \mathcal{K} \cup \{\mathsf{Target} \equiv C\}$ we have $\mathcal{K}' \models E_p$ and $\mathcal{K}' \not\models E_n$. This problem can be solved through a search in the space of concepts. Lehmann and Hitzler [13] impose an ordering on this search space and then use *refinement operators* to traverse it. Intuitively, downward (upward) refinement operators construct specializations (generalizations) of hypotheses. De Raedt [7] defines three settings for learning a hypothesis in ILP:

i) learning from entailments: the examples are definite clauses and a hypothesis H covers an example e with background knowledge B if and only if $B \cup H \models e$. In many well-known systems, such as FOIL [15], one requires that the examples are ground facts, a special case of definite clauses.
ii) learning from interpretations: the examples are *Herbrand interpretations* (set of true ground facts that completely describe a possible situation) and an hypothesis H covers an example e with background knowledge B if and only if e is a model of $B \cup H$.
iii) learning from proofs: the examples are ground proof-trees and an example e is covered by a hypothesis H with background knowledge B if and only if e is a proof-tree for $H \cup B$.

The key difference between learning from interpretations and learning from entailment is that interpretations carry much more information. Indeed, when learning from entailments, an example may consist of a single fact, whereas when learning from interpretations, complete descriptions must be available. Therefore, learning from interpretations requires less computational effort than learning from entailments [6]. Another point is that learning from interpretations is well suited for learning from positive examples only.

DL learning algorithms such as DL-Learner and DL-FOIL adopt learning from entailments. In that case, examples are assertions about the new concept Target and an example e is covered by a knowledge base $\mathcal{K}$ if and only if $\mathcal{K} \models e$. A DL learning algorithm can be proposed based on learning from interpretations setting, as we propose in the next section for the PDL CR$\mathcal{ALC}$.

3 Learning in CR$\mathcal{ALC}$

In this section, we propose our algorithm for probabilistic inclusion learning in CR$\mathcal{ALC}$. We assume learning from interpretations, so as to handle gracefully the

interpretation-based semantics of CR$\mathcal{ALC}$. The idea is as follows. Suppose that a concept C is learned (logically only, without probabilities!) as $C \equiv D \sqcap F$. However, there is noise in the data and a probabilistic inclusion relating these concepts would be interesting. Therefore, probabilistic inclusions of the form $P(C|X) = \alpha$ are examined, where X are concepts based on D and F, and α is a constant learned by maximum likelihood estimation. Here we search for the best conditioning concept, by considering parts of the concept definition; in our example, X can be the whole domain, D, F or D $\sqcap F$, therefore, probabilities such as $P(C)$, $P(C|D)$, $P(C|F)$ and $P(C|D \sqcap F)$ are evaluated. We then examine, based on an evaluation function, whether it is best to discard the logical definition (that is, $C \equiv D \sqcap F$). In short, we focus on learning probabilistic inclusions given a base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a set of examples E. We assume that a description logic learning algorithm, for instance DL-Learner or DL-FOIL, is first run to learn definitions for concepts in $\mathcal{T}$.

3.1 Direct Inference and EM

As CR$\mathcal{ALC}$ adopts an interpretation-based semantics, we would ideally need data on each individual in order to learn descriptions. For instance, if we wanted to learn the probability of $P(\mathsf{Professor}(a))$, we would need a dataset with information about the object a having or not the property Professor. Different objects can have different probabilities; most probabilistic description logics that adopt interpretation-based semantics combine these different probabilities into a single value to be adopted by all objects (that is, $\forall x : P(C(x)) = \alpha$).

Most available datasets contain information about all (or at least many) objects of the domain, instead of information about one specific object. Learning with such a dataset means learning probabilities with domain-based semantics. In this paper we consider that the probabilities learned with domain-based semantics are a good approximation for probabilities with interpretation-based semantics; that is, we conduct *direct inference* by transfering learned probabilities from domain to interpretations. Our algorithm learns probabilities in probabilistic inclusions by approximating the probabilities estimated with domain-based semantics.

For estimating domain-based probabilities, we resort to the *Expectation Maximization (EM)* algorithm [8]. The likelihood function (the *score*) to be maximized is

$$L = \prod_{e \in E} P(e|\mathcal{K}).$$

3.2 Learning the Structure of Concepts

Our algorithm for structure learning starts by taking concept names as probabilistically independent, and by learning probabilistic inclusions such as $P(C)$. Then the algorithm verifies whether a concept name provides better results (better inference) if probabilistically conditioned on another concept name or on a

concept description. For that the algorithm considers possibilities from the logical definition of the concept (as learned by ILP), since it seems nature that the concept C can only be conditioned on concepts that the learning process learned that it is related with. The algorithm uses subsumption, and keeps only the conditioning concept that provides the highest score. When a candidate probabilistic inclusion uses only part of a original logically learned concept description, further analysis is needed. A decision must be taken as to whether the probabilistic inclusion should be added and the original concept should be discarded. (Note that if both the original concept description and the probabilistic inclusion were kept together, then we might not be able to construct a unique probability distribution out of the learned probabilistic terminology). The procedure is detailed in Algorithm 1.

Algorithm 1. Algorithm for learning probabilistic inclusions

Require: a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a training set E.
Ensure: a new knowledge base $\mathcal{K} =< \mathcal{T}_f, \mathcal{A} >$ with probabilistic inclusions
1: $\mathcal{T}_f = \mathcal{T}$
2: calculate score $score_f$ using E;
3: **for all** concept $C \in \mathcal{T}$ **do**
4: search in $\mathcal{T}$ for the definition of concept C and include its parts in a set H;
5: **if** concept C is subsumed by some other concept D **then**
6: include D in H;
7: **end if**
8: $score_c = score_f$;
9: initialize A_c;
10: **for all** $h \in H$ **do**
11: set $\mathcal{T}_h = \mathcal{T} \cup P(C|h)$;
12: learn probabilities using E;
13: calculate score $score_h$ using E;
14: **if** $score_h > score_c$ **then**
15: $score_c = score_h$
16: $A_c = P(C|h)$
17: **end if**
18: **end for**
19: $\mathcal{T}_f = \mathcal{T}_f \cup A_c$ (insertion restrictions are examined).
20: **end for**

Algorithm 1 receives a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a training set E, composed by interpretations. For each concept in the terminology $\mathcal{T}$, the algorithm searches for its definitions. All subsumed concepts of this definition are considered separately and included in the set H (line 4); the empty and the proper definition are also inserted in H. If an inserted concept is subsumed by another concept, this latter concept is also included in H (line 6). The variable $score_c$ is initialized with the score of $\mathcal{K}$ (line 8), and a variable A_c is initialized as empty (line 9). Each concept included in H is considered separately as

conditioning the concept of interest (line 11). The probability of this new probabilistic inclusion is estimated using the EM algorithm and then the new score is calculated (lines 12 e 13). If this new score improves the current one, the probabilistic inclusion is taken as the current best one and is kept in A_c (line 16). After learning a probability inclusion for each concept the algorithm ends, returning a knowledge base with a new terminology. If the initial score is not improved by neither probability inclusion proposed for a concept, this concept is not associated with any probabilistic inclusion.

After the learning algorithm is run, a pruning routine can verify whether the terminology can be simplified. For example, the inclusion $\mathsf{Researcher} \sqsubseteq \mathsf{Professor}$ can be automatically deleted from the terminology in the presence of probabilistic inclusions $P(\mathsf{Researcher}|\mathsf{Professor}) = 0.4$ and $P(\mathsf{Researcher}|\neg\mathsf{Professor}) = 0.0$.

4 Preliminary Results

Experiments have been run on data extracted from the Lattes Curriculum Platform, a public repository containing data about Brazilian researchers, such as name, address, education, professional experience, areas of expertise. Bibliographic output and participation on examination boards are our main interest. For these experiments, concept descriptions were learned using DL-Learner, available at http://dl-learner.org/Projects/DLLearner. Our algorithm was implemented in the Java language, running in an Ubuntu Linux system with 4GB RAM, 2.4GHZ INTEL CORE 2 DUO. We selected 202 researchers randomly and extracted data such as:

$\mathsf{Researcher}(\mathsf{r1}), \mathsf{Researcher}(\mathsf{r2}), \mathsf{Researcher}(\mathsf{r4}), \ldots$
$\mathsf{wasAdvised}(\mathsf{r8}, \mathsf{r179}), \mathsf{wasAdvised}(\mathsf{r30}, \mathsf{r83}), \mathsf{wasAdvised}(\mathsf{r33}, \mathsf{r1}), \ldots$
$\mathsf{sharePublication}(\mathsf{r1}, \mathsf{r32}), \mathsf{sharePublication}(\mathsf{r4}, \mathsf{r12}), \mathsf{sharePublication}(\mathsf{r5}, \mathsf{r115}), \ldots$
$\mathsf{sameExaminationBoard}(\mathsf{r1}, \mathsf{r32}), \mathsf{sameExaminationBoard}(\mathsf{r4}, \mathsf{r12}), \ldots$
$\mathsf{hasSameInstitution}(\mathsf{r1}, \mathsf{r27}), \mathsf{hasSameInstitution}(\mathsf{r1}, \mathsf{r28}), \ldots$
$\mathsf{advises}(\mathsf{r1}, \mathsf{r33}), \mathsf{advises}(\mathsf{r1}, \mathsf{r171}), \mathsf{advises}(\mathsf{r1}, \mathsf{r81}), \ldots$

We aimed at learning a probabilistic terminology that best describes a set of collaboration patterns among researchers. Learning was performed in two steps. The first step was concerned with learning definitions for concepts (such as NearCollaborator, NullMobilityResearcher, StrongRelatedResearcher) given positive and negative examples. For instance, the pair of individuals r_1 and r_4 could be positive examples of the NearCollaborator concept. Intuitively, a near collaborator is a researcher who shares a publication with a colleague of the same institution. On the other hand, an institution can be interested in asking about mobility of their former students; that is, whether they remain as staff members after finishing their PhDs or not. The concept NullMobilityResearcher is intended to define a researcher with null mobility. The concepts StrongRelatedResearcher and InheritedResearcher define categories of relationships among researchers.

As noted previously, DL-Learner was employed to generate initial concepts based on background knowledge and examples. A preliminary terminology is:

NearCollaborator $\equiv$ Researcher
$\sqcap$ ($\exists$hasSameInstitution.$\exists$sharePublication.
$\exists$sharePublication.$\exists$advises.Researcher) $\sqcap$
($\exists$sharePublication.$\exists$hasSameInstitution.
$\exists$sharePublication.Researcher)

FacultyNearCollaborator $\equiv$ NearCollaborator
$\sqcap$ $\exists$sameExaminationBoard.Researcher

NullMobilityResearcher $\equiv$ Researcher
$\sqcap$($\exists$hasSameInstitution.$\exists$sameExaminationBoard.
$\exists$wasAdvised.$\forall$advises.$\exists$hasSameInstitution.Researcher) $\sqcap$
($\exists$wasAdvised.$\exists$hasSameInstitution.Researcher)

StrongRelatedResearcher $\equiv$ Researcher
$\sqcap$ ($\exists$sharePublication.Researcher $\sqcap$
$\exists$wasAdvised.Researcher)

InheritedResearcher $\equiv$ Researcher
$\sqcap$ ($\exists$sameExaminationBoard.Researcher $\sqcap$
$\exists$wasAdvised.Researcher).

The next step focused on learning probabilistic inclusions; Following concept definitions, possible probabilistic inclusions for concept NearCollaborator are:

P(NearCollaborator|Researcher $\sqcap$ ($\exists$hasSameInstitution.$\exists$sharePublication.
$\exists$sharePublication.$\exists$advises.Researcher) $\sqcap$ ($\exists$sharePublication.
$\exists$hasSameInstitution.$\exists$sharePublication.Researcher)) $= \alpha$,

P(NearCollaborator|Researcher) $= \beta$,

P(NearCollaborator|Researcher $\sqcap$ $\exists$hasSameInstitution.
$\exists$sharePublication.$\exists$sharePublication.$\exists$advises.Researcher) $= \gamma$

P(NearCollaborator|Researcher $\sqcap$ $\exists$sharePublication.
$\exists$hasSameInstitution.$\exists$sharePublication.Researcher) $= \theta$.

Based on a probabilistic score (maximum likelihood on the set of examples), the last candidate was chosen. This choice implies that the original concept description must be removed. The complete probabilistic terminology is:

P(Researcher) $= 1.0$ P(wasAdvised) $= 0.29$
P(hasSameInstitution) $= 0.83$ P(sharePublication) $= 0.73$
P(sameExaminationBoard) $= 0.41$

P(NearCollaborator | Researcher $\sqcap$ $\exists$sharePublication.$\exists$hasSameInstitution.
$\exists$sharePublication.Researcher) $= 0.95$
FacultyNearCollaborator $\equiv$ NearCollaborator
$\sqcap$ $\exists$sameExaminationBoard.Researcher

P(NullMobilityResearcher | Researcher $\sqcap$ $\exists$wasAdvised.
$\exists$hasSameInstitution.Researcher) $= 0.98$
StrongRelatedResearcher $\equiv$ Researcher
$\sqcap$ ($\exists$sharePublication.Researcher $\sqcap$
$\exists$wasAdvised.Researcher)

InheritedResearcher $\equiv$ Researcher
$\sqcap$ ($\exists$sameExaminationBoard.Researcher $\sqcap$
$\exists$wasAdvised.Researcher)

We can observe that both StrongRelatedResearch and InheritedResearcher have no probabilistic inclusions. Conversely, NearCollaborator and NulleMobilityResearcher concepts have been replaced by suitable shorter probabilistic inclusions.

To investigate the application of learned concepts, some inferences in this resulting CR$\mathcal{ALC}$ terminology were calculated using an exact (propositionalization) algorithm [3]. When Researcher(0) and Researcher(1)[1] are given as evidence, we obtain $P(\mathsf{NullMobilityResearcher}(0)) = 0.46$. Further evidence changes this value to:

$$P(\mathsf{NullMobilityResearcher}(0)|\exists \mathsf{wasAdvised}(0,1)) = 0.83,$$

and to:

$$P(\mathsf{NullMobilityResearcher}(0)|\exists \mathsf{wasAdvised}(0,1) \sqcap \exists \mathsf{hasSameInstitution}(0,1)) = 0.9.$$

This last probability value indicates that individual 0 can be placed within NullMobilityResearcher. (Note that given the same evidence, the originally learned deterministic ontology would misclassify the individual 0.)

5 Conclusion

We have presented techniques for learning concepts in a probabilistic description logic from relational data. Learning occurs in two steps. In the first step, concept descriptions are learned by existing description logic learning algorithms. The second step searches for probabilistic inclusions that can improve the logical descriptions. The algorithm actually compares logical and probabilistic inclusions, keeping the most accurate ones and discarding the others (where performance is measured using likelihood). The idea is that by letting probabilities leave descriptions that are a little more flexible, we can obtain better models for dealing with real data. Experiments, focused on learning a probabilistic terminology from a real-world domain (the Lattes scientific repository), suggest that probabilistic inclusions do lead to improved likelihoods.

Probabilistic description logics offer expressive languages in which to conduct learning, while charging a relatively low cost for inference. The present contribution offers novel ideas for this sort of learning task; we note that the current literature on this topic is rather scarce. Our future work is to investigate the scalability of our learning methods, with an interest in applications for the semantic web.

Acknowledgements

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

[1] In these calculations, indexes $0, 1 \ldots n$ represent individuals.

References

1. Antoniou, G., van Harmelen, F.: Semantic Web Primer. MIT Press, Cambridge (2008)
2. Baader, F., Nutt, W.: Basic description logics. In: Description Logic Handbook, pp. 47–100. Cambridge University Press, Cambridge (2002)
3. Cozman, F.G., Polastro, R.B.: Loopy propagation in a probabilistic description logic. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 120–133. Springer, Heidelberg (2008)
4. Cozman, F.G., Polastro, R.B.: Complexity analysis and variational inference for interpretation-based probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence (2009)
5. De Raedt, L. (ed.): Advances in Inductive Logic Programming. IOS Press, Amsterdam (1996)
6. De Raedt, L.: Logical settings for concept-learning. Artificial Intelligence 95(1), 187–201 (1997)
7. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S.H. (eds.): Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911. Springer, Heidelberg (2008)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal Royal Statistical Society B 44, 1–38 (1977)
9. Fanizzi, N., D'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
10. Heinsohn, J.: Probabilistic description logics. In: International Conf. on Uncertainty in Artificial Intelligence, pp. 311–318 (1994)
11. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Principals of Knowledge Representation (KR), pp. 461–472 (1994)
12. Jaeger, M.: Relational bayesian networks: a survey. Linkoping Electronic Articles in Computer and Information Science 6 (2002)
13. Lehmann, J.: Hybrid learning of ontology classes. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 883–898. Springer, Heidelberg (2007)
14. Ochoa-Luna, J.E., Cozman, F.G.: An algorithm for learning with probabilistic description logics. In: 5th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 8th International Semantic Web Conference (ISWC), Chantilly, USA, pp. 63–74 (2009)
15. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
16. Sebastiani, F.: A probabilistic terminological logic for modelling information retrieval. In: ACM Conf. on Research and Development in Information Retrieval (SIGIR), pp. 122–130 (1994)