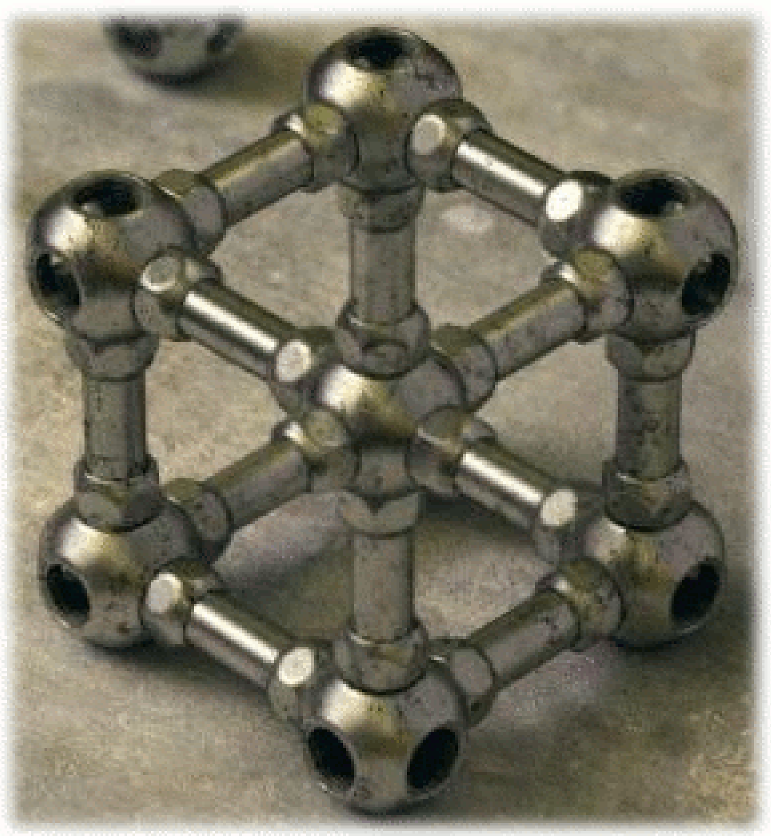

An Algorithm for Learning with Probabilistic Description Logics [1]

Author(s):

José Eduardo Ochoa Luna

Fabio Gagliardi Cozman

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

An Algorithm for Learning with Probabilistic Description Logics

José Eduardo Ochoa Luna and Fabio Gagliardi Cozman

Escola Politécnica, Universidade de São Paulo,
Av. Prof. Mello Morais 2231, São Paulo - SP, Brazil
`eduardo.ol@gmail.com, fgcozman@usp.br`

Abstract. Probabilistic Description Logics are the basis of ontologies in the Semantic Web. Knowledge representation and reasoning for these logics have been extensively explored in the last years; less attention has been paid to techniques that learn ontologies from data. In this paper we report on algorithms that learn probabilistic concepts and roles. We present an initial effort towards semi-automated learning using probabilistic methods. We combine ILP (Inductive Logic Programming) methods and a probabilistic classifier algorithm (search for candidate hypotheses is conducted by a Noisy-OR classifier). Preliminary results on a real world dataset are presented.

1 Introduction

Ontologies are key components of the Semantic Web, and among the formalisms proposed within Knowledge Engineering, the most popular ones at the moment are based on Description Logics (DLs) [1]. There are however relatively few ontologies available, and on very few subjects. Moreover, building an ontology from scratch can be a very burdensome and difficult task; very often two domain experts design rather different ontologies for the same domain [2]. Considerable effort is now invested into developing automated means for the acquisition of ontologies. Most of the currently pursued approaches do not use the expressive power of languages such as OWL, and are only capable of learning ontologies of restricted form, such as taxonomic hierarchies [3].

It is therefore natural to try to combine logic-based and probabilistic approaches to machine learning for automated ontology acquisition. Inspired by the success of Inductive Logic Programming (ILP) [4] and statistical machine learning, in this paper we describe methods that learn ontologies in the recently proposed Probabilistic Description Logic CR$\mathcal{ALC}$ [5]. In using statistical methods we wish to cope with the uncertainty that is inherent to real-world knowledge bases, where we commonly deal with biased, noisy or incomplete data. Moreover, many interesting research problems, such as ontology alignment and collective classification, require probabilistic inference over evidence.

Probabilistic Description Logics are closely related to Probabilistic Logics that have been extensively researched in the last decades [6, 7]. Some logics [8] admit inference based on linear programming, while other resort to independence assumptions and graph-theoretical models akin to Bayesian and Markov networks. Despite the potential applicability of Probabilistic Description Logics, learning ontologies expressed in these logics is a topic that has not received due attention. In principle, it might be argued that the same methods for learning probabilistic logics might be applied, with proper account of differences in syntax and semantics. In this paper we follow this path and report on some similarities and differences that may be of interest.

The semantics of the Probabilistic Description Logic CR$\mathcal{ALC}$ [5] is based on measures over interpretations and on assumptions of independence. Scalability issues for inference in CR$\mathcal{ALC}$ have been addressed so that we can run inference on medium size domains [9]. There are two learning tasks that deserve attention. First, learning *probability values*, perhaps through a maximum likelihood estimation method. We use this technique and, due to uncertainty in Semantic Web datasets, we employ the EM algorithm. The second task is learning *logical constructs*, where we are interested in finding a set of concepts and roles that best fit examples and where probabilistic assessments can be assigned. In ILP algorithms such as FOIL [10], one commonly relies on a cover function to evaluate candidate hypotheses. We approach learning concepts as a classification task, and based on an efficient probabilistic Noisy-OR classifier [11, 12], we guide the search among candidate structures.

Section 2 reviews key concepts useful to the remainder of the paper. In Section 3, algorithms for learning CR$\mathcal{ALC}$ ontologies are introduced. Once these algorithms have formally stated, we wish to explore semi-automated reasoning from a real world dataset — the Lattes curriculum platform. A first attempt at constructing a probabilistic ontology using this dataset is reported in Section 4.

2 Background

Assume we are provided with a repository of HTML pages where researchers and students have stored data about publications, courses, languages, and further relational data. In order to structure such knowledge we might choose to use ontologies. We may extract *concepts* such as $\mathsf{Researcher}$ and Person, and we may establish relationships such as $\sqsubseteq$ among them. These concepts are often expressed using Description Logics (Section 2.1). Suppose we are not able to precisely state membership relations among concepts, but instead we can give probabilistic assessments such as $P(\mathsf{Student}|\mathsf{Researcher}) = 0.4$. Such assessments are encoded in Probabilistic Description Logics such as CR$\mathcal{ALC}$ (Section 2.2). Suppose further that we look for automated means to learn ontologies given *assertions on concepts* such as $\mathsf{Student}(\mathsf{jane})$; this task is commonly tackled by Description Logic Learning algorithms (Section 2.3).

2.1 Description Logics

Description Logics (DLs) form a family of representation languages that are typically decidable fragments of First Order Logic (FOL) with particular semantics [13, 14]. DLs represent knowledge in terms of *objects*, *concepts*, and *roles*. Each *concept* in the set of concepts $N_{\mathsf{C}} = \{\mathsf{C}, \mathsf{D}, \ldots\}$ is interpreted as a subset of a domain (a set of objects). Each *role* in the set of roles $N_{\mathsf{R}} = \{\mathsf{r}, \mathsf{s}, \ldots\}$ is interpreted as a binary relation on the domain. Individuals represent the objects through names from the set of names $N_{\mathcal{I}} = \{\mathsf{a}, \mathsf{b}, \ldots\}$. Information is stored in a knowledge base divided in (at least) two parts: the TBox (terminology) and the ABox (assertions). The TBox describes the terminology by listing concepts and roles and their relationships. The ABox contains assertions about objects.

Complex concepts are built using atomic concepts, roles and constructors. Depending on the constructors involved one can obtain different expressive power and decidability properties. The semantics of a description is given by a *domain* Δ and an *interpretation*, that is a functor $\cdot^{\mathcal{I}}$. We refer to [13] for further background on Description Logics.

One of the central ideas in DL is *subsumption* [14]: Given two concepts descriptions C and D in $\mathcal{T}$, C subsumes D denoted by $\mathsf{C} \sqsupseteq \mathsf{D}$, iff for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathsf{C}^{\mathcal{I}} \supseteq \mathsf{D}^{\mathcal{I}}$. Also, $\mathsf{C} \equiv \mathsf{D}$ amounts to $\mathsf{C} \sqsupseteq \mathsf{D}$ and $\mathsf{D} \sqsupseteq \mathsf{C}$.

Subsumption is a useful inference mechanism that allow us to perform standard reasoning tasks such as *instance checking* and *concept retrieval.* Instance checking is valuable for our ILP methods because it amounts to produce class-membership assertions: $\mathcal{K} \models \mathsf{C}(\mathsf{a})$, where $\mathcal{K}$ is the knowledge base, a is an individual name and C is a concept definition given in terms of the concepts accounted for in $\mathcal{K}$.

2.2 The Logic CR$\mathcal{ALC}$

The logics mentioned in Section 2.1 do not handle uncertainty through probabilities. It might be interesting to assign probabilities to assertions, concepts, roles; the Probabilistic Description Logic CR$\mathcal{ALC}$ does just that.

CR$\mathcal{ALC}$ is a probabilistic extension of the DL $\mathcal{ALC}$. The following constructors are available in $\mathcal{ALC}$: *conjunction* $(\mathsf{C} \sqcap \mathsf{D})$, *disjunction* $\mathsf{C} \sqcup \mathsf{D}$, *negation* $(\neg \mathsf{C})$, *existential* restriction $(\exists r.\mathsf{C})$, and value restriction $(\forall r.\mathsf{C})$. Concepts inclusions and definitions are allowed and denoted by $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{C} \equiv \mathsf{D}$, where C is a concept name. The semantics is given by a domain $\mathcal{D}$ and an interpretation $\mathcal{I}$. A set of concept inclusions and definitions is a terminology. A terminology is acyclic if it is a set of concept inclusions/definitions such that no concept in the terminology uses itself.

A key concept in CR$\mathcal{ALC}$ is *probabilistic inclusion*, denoted by $P(\mathsf{C}|\mathsf{D}) = \alpha$, where D is a concept and C is a concept name. If the interpretation of D is the whole domain, then we simply write $P(\mathsf{C}) = \alpha$. We are interested in computing a *query* $P(\mathsf{A}_o(\mathsf{a}_0)|\mathcal{A})$ for an ABox $\mathcal{A} = \{\mathsf{A}_j(\mathsf{a}_j)\}_{j=1}^{M}$ (this is an *inference*). Assume also that C in role restrictions $\exists\mathsf{r}.\mathsf{C}$ and $\forall\mathsf{r}.\mathsf{C}$ is a concept name. As probabilistic inclusions must only have concept names in their conditioned concept, assessments such as $P(\forall\mathsf{r}.\mathsf{C}|\mathsf{D}) = \alpha$ or $P(\exists\mathsf{r}.\mathsf{C}|\mathsf{D}) = \alpha$ are not allowed.

We assume that every terminology is acyclic; this assumption allows one to draw any terminology $\mathcal{T}$ as a directed acyclic graph $\mathcal{G}(\mathcal{T})$: each concept name is a node, and if a concept C directly uses concept D, then D is a *parent* of C in $\mathcal{G}(\mathcal{T})$. Each existential and value restriction is added to the graph $\mathcal{G}(\mathcal{T})$. As each one of these restrictions directly uses r and C, the graph must contain a node for each role r, and 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 can be computed by a first order loopy propagation algorithm that has been shown to produce good approximations for medium size domains [9].

2.3 Inductive Logic Programming and Description Logic Learning

ILP is a research field at the intersection of machine learning [15] and logic programming [16]. It aims at a formal framework as well as practical algorithms for inductively learning relational descriptions from examples and background knowledge. Learning is commonly regarded as a search problem [17]; indeed, in ILP there is a space of candidate solutions, the set of "well formed" hypotheses **H**, and an acceptance criterion characterizing solutions. In concept-learning and ILP the search space is typically structured by means of the dual notions of generalization and specialization.

A significant algorithm for ILP is FOIL [10]. This algorithm moves from an explicit representation of the target relation (as a set of tuples of a particular collection of constants) to a more general, functional definition that might be applied to different constants. For a particular target relation, FOIL finds clauses in FOL one at a time, removing tuples explained by the current clause before looking for the next through a *cover* method. FOIL uses an information-based heuristic to guide its search for simple, general clauses. Because of its simplicity and computational efficiency, we have chosen to develop a covered approach when learning Probabilistic Description Logics.

There have been notable efforts to learn ontologies in Description Logics; some of these previous results have directly inspired our work. As noted by Fanizzi et al [14], early work on learning in DLs essentially focused on demonstrating the PAC-learnability for various languages derived from CLASSIC. Many approaches to the problem of learning concept definitions in DL formalisms can be classified in two categories [2]: in one category the problem is approached by translating it to another formalism in which concept learning has already been investigated, while in the other category the problem is approached in the original formalism.

One example of the first approach can be found in the work of Kietz [18], where the hybrid language CARIN-$\mathcal{ALN}$ is used [19]. This language combines a complete structural subsumption service in a DL with Horn logic, where terms are individuals in the domain. Likewise, in the $\mathcal{AL}$-log framework [20], DATALOG clauses are combined with $\mathcal{ALC}$ constructs. In the same direction, $\mathcal{DL}$+log [21] allows for the tight integration of DLs and DATALOG. Arguably, the decidable knowledge representation framework $\mathcal{SHIQ}$+log [1], is the most powerful among the ones currently available for the integration of DLs and clausal logic. First, it relies on the very expressive DL $\mathcal{SHIQ}$. Second, it allows for inducing a definition for DL concepts thus having ontology elements not only as input but also as output of the learning process.

The problem of translation turns out to be similar to an ILP problem. There are two issues to address: incompatibilities between DLs and Horn Logic, and the fact that the OWA[1] is used in DLs.

The other approach, solving the learning problem in the original formalism, can be found in the work of Cohen and Hirsh [22], which uses a pure DL-based approach for concept learning, in this case on the CLASSIC DL language. In these algorithms, ILP has been a significant influence, as refinement operators have been extensively explored. Badea and Nienhuys-Cheng [23] suggest a refinement operator for the $\mathcal{ALER}$ description logic. They also investigate some theoretical properties of refinement operators that favour the use of a downward refinement operator to enable a top-down search.

Learning algorithms for DLs (in particular for the language $\mathcal{ALC}$) were created by Iannone et al [2] that also make use of refinement operators. Instead of using the classical approach of combining refinement operators with a search heuristic, they developed an example driven learning method. The language, called YINYANG, requires lifting the instances to the concept level through a suitable approximate operator (most specific concepts MSCs) and then start learning from such extremely specific concept descriptions. A problem of these algorithms is that they tend to produce unnecessarily long concepts. One reason is that MSCs for $\mathcal{ALC}$ and more expressive languages do not exist and hence can only be approximated.

These disadvantages have been partly mitigated in the work of Lehmann [24], where approximations are not needed because it is essentially based on a genetic programming procedure lying on refinement operators whose fitness is computed on the grounds of the covered instances. In the DL-LEARNER system [3] further refinement operators and heuristics have been developed for the $\mathcal{ALC}$ logic.

The DL-FOIL system [14] is a new DL version of the FOIL [10] algorithm, that is adapted to learning the DL representations supporting the OWL-DL language. The main components of this new system are represented by a set of refinement operators borrowed from other similar systems [2, 3] and by a different gain function (proposed in FOIL-I [25]) which must take into account the OWA inherent to DLs. In DL-FOIL, like in the original FOIL algorithm, the generalization routine computes (partial) generalizations as long as they do not cover any negative example. If this occurs, the specialization routine is invoked for solving these sub-problems. This routine applies the idea of specializing using the (incomplete) refinement operator. The specialization continues until no negative example is covered (or a limited number of them).

[1] Open World Assumption mantains that an object that cannot be proved to belong to a certain concept is not necessarily a counterexample for that concept [14].

3 Probabilistic Description Logic Learning

In this section, we focus on learning DL axioms and probabilities tailored to CR$\mathcal{ALC}$. To learn the terminology component we are inspired by Probabilistic ILP methods and thus we follow generic syntax and semantics given in [16]. The generic supervised concept learning task is devoted to finding axioms that best represent assertions positive (covered) and negatives, in a probabilistic setting this *cover relation* is given by:

Definition 1. (Probabilistic Covers Relation) *A probabilistic covers relation takes as arguments an example e, a hypothesis H and possibly the background theory B, and returns the probability value $P(e|H, B)$ between 0 and 1 of the example e given H and B, i.e., $covers(e, H, B) = P(e|H, B)$.*

Given Definition 1 we can define the Probabilistic DL learning problem as follows [16]:

Definition 2. (The Probabilistic DL Learning Problem) *Given a set $E = E_p \cup E_i$ of observed and unobserved examples E_p and E_i (with $E_p \cap E_i = \emptyset$) over the language $\mathcal{L}_E$, a probabilistic covers relation $covers(e, H, B) = P(e|H, B)$, a logical language $\mathcal{L}_H$ for hypotheses, and a background theory B, find a hypothesis H^* such that $H^* = \arg\max_H score(E, H, B)$ and the following constraints hold: $\forall e_p \in E_p : covers(e_p, H^*, B) > 0$ and $\forall e_i \in E_i : covers(e_i, H^*, B) = 0$. The score is some objective function, usually involving the probabilistic covers relation of the observed examples such as the observed likelihood $\prod_{e_p \in E_p} covers(e_p, H^*, B)$ or some penalized variant thereof.*

Negative examples conflict with the usual view on learning examples in statistical learning. Therefore, when we speak of positive and negative examples we are referring to observed and observed ones.

As we focus in CR$\mathcal{ALC}$, $B = \mathcal{K} = (\mathcal{T}, \mathcal{A})$, and given a target concept C, $E = Ind^+_{\mathsf{C}}(\mathcal{A}) \cup Ind^-_{\mathsf{C}}(\mathcal{A}) \sqsupseteq Ind(\mathcal{A})$, are positive and negative examples or individuals. For instance, candidate hypotheses can be given by $\mathsf{C} \sqsupseteq H_1, \ldots, H_k$, where $H_1 = \mathsf{B} \sqcap \exists \mathsf{D}.\top, H_2 = \mathsf{A} \sqcup \mathsf{E}, \ldots$.

We assume each candidate hypothesis together with the example e for the target concept as being a probabilistic variable or feature in a probabilistic model[2]; according to available examples, each candidate hypothesis turns out to be true, false or unknown whether result for instance checking $\mathsf{C}(\mathsf{a})$ on $\mathcal{K}, Ind(\mathcal{A})$ is respectively true, false or unknown. The learning task is restricted to finding a probabilistic classifier for the target concept.

A suitable framework for this probabilistic setting is the Noisy-OR classifier, a probabilistic model within the Bayesian networks classifiers commonly referred to as models of independence of clausal independence (ICI) [12]. In a Noisy-OR classifier we aim at learning a class C given a large number of attributes.

As a rule, in an ICI classifier for each attribute variable $A_j, j = 1, \ldots, k$ ($\mathbf{A}$ denotes the multidimensional variable $(A_1, \ldots, A_k)$ and $\mathbf{a} = (a_1, \ldots, a_k)$

[2] A similar assumption is adopted in the nFOIL algorithm [26].

its states) we have one child A'_j that has assigned a conditional probability distribution $P(A'_j|A_j)$. Variables $A'_j, j = 1, \ldots, k$ are parents of probability of the class variable C. $P_M(C|\mathbf{A}')$ represents a deterministic function f that assigns to each combination of values $(a'_1, \ldots, a'_k)$ a class c. A generic ICI classifier is illustrated in Figure 1 .

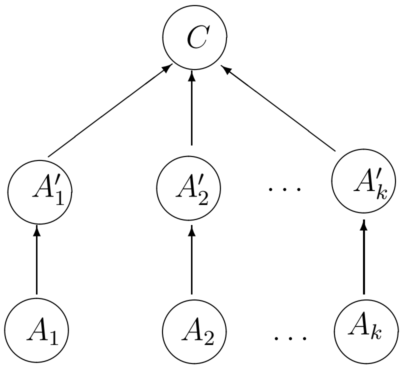

Fig. 1. ICI models [12].

The probability distribution of this model is given by [12]:

$$P_M(c, \mathbf{a}', \mathbf{a}) = P_M(c|\mathbf{a}') \prod_{j=1}^{k} P_M(a'_j|a_j) \cdot P_M(a_j),$$

where the conditional probability $P_M(c|\mathbf{a}')$ is one if $c = f(\mathbf{a}')$ and zero otherwise. The Noisy-OR model is an ICI model where f is the OR function:

$$P_M(C = 0|\mathbf{A}' = 0) = 1 \text{ and } P_M(C = 0|\mathbf{A}' \neq 0) = 0.$$

The joint probability distribution of the Noisy-OR model is

$$P_M(\cdot) = P_M(C|A'_1, \ldots, A'_k) \cdot \left(\prod_{j=1}^{k} P_M(A'_j|A_j) \cdot P_M(A_j) \right).$$

It follows that

$$P_M(C = 0|\mathbf{A} = \mathbf{a}) = \prod_j P_M(A'_j = 0|A_j = a_j), \tag{1}$$

$$P_M(C = 1|\mathbf{A} = \mathbf{a}) = 1 - \prod_j P_M(A'_j = 0|A_j = a_j). \tag{2}$$

Using a threshold $0 \leq t \leq 1$ all data vectors $\mathbf{a} = (a_1 \ldots, a_k)$ such that $P_M(C = 0|\mathbf{A} = \mathbf{a}) < t$ are classified to class $C = 1$.

The Noisy-OR classifier has the following semantics. If an attribute A_j is in a state a_j then the instance $(a_1, \ldots, a_j, \ldots, a_k)$ is classified as $C = 1$ unless there is an inhibitory effect, with probability $P_M(A'_j = 0|A_j = a_j)$. All inhibitory effects are assumed to be independent. Therefore the probability that an instance does not belong to class C ($C = 0$), is a product of all inhibitory effects $\prod_j P_M(A'_j = 0|A_j = a_j)$. For learning this classifier the EM-algorithm has been proposed [12]. The algorithm is directly applicable to any ICI model; in fact, an efficient implementation resort to a transformation of an ICI model using a hidden variable (further details in [12]). We now shortly review the EM-algorithm tailored to Noisy-OR combination functions.

Every iteration of the EM-algorithm consists of two steps: the expectation step (E-step) and maximization step (M-step). In a transformed decomposable model the E-step corresponds to computing the expected marginal count $n(A'_l, A_l)$ given data $D = \{\mathbf{e}^1, \ldots, \mathbf{e}^n\}$ ($\mathbf{e}^i = \{c^i, \mathbf{a}^i\} = \{c^i, a^i_1, \ldots, a^i_k\}$) and model M:

$$n(A'_l, A_l) = \sum_{i=1}^{n} P_M(A'_l, A_l|\mathbf{e}^i) \text{ for all } l = 1, \ldots, k$$

where for each (a'_l, a_l)

$$P_M(A'_l = a'_l, A_l = a_l|\mathbf{e}^i) = \begin{cases} P_M(A'_l = a'_l|\mathbf{e}^i) & \text{if } a_l = a^i_l, \\ 0 & \text{otherwise.} \end{cases}$$

Assume a Noisy-Or classifier P_M and an evidence $C = c, \mathbf{A} = \mathbf{a}$. The updated probabilities (the E-step) of A'_l for $l = 1, \ldots, k$ can be computed as follows [12]:

$$P_M(A'_l = a'_l|C = c, \mathbf{a})$$

$$= \begin{cases} 1 & \text{if } c = 0 \text{ and } a'_l = 0, \\ 0 & \text{if } c = 0 \text{ and } a'_l = 1, \\ \frac{1}{z} \cdot \left(P_M(A'_l = 0|A_l = a_l) - \prod_j P_M(A'_j = 0|A_j = a_j) \right) & \text{if } c = 1 \text{ and } a'_l = 0, \\ \frac{1}{z} \cdot P_M(A'_l = 1|A_l = a_l) & \text{if } c = 1 \text{ and } a'_l = 1, \end{cases}$$

where z is a normalization constant. The maximization step corresponds to setting

$$P^*_M(A'_l|A_l) = \frac{n(A'_l, A_l)}{n(A_l)}, \text{ for all } l = 1 \ldots, k.$$

Given the Noisy-OR classifier, the complete learning algorithm is described in Figure 2, where λ denotes the maximum likelihood parameters. We have used the refinement operators introduced in [3] and the Pellet reasoner[3] for instance checking. It may happen that during learning a given example for a candidate hypothesis H_i cannot be proved to belong to the target concept. This is not necessarily a counterexample for that concept. In this case, we can make use of

[3] http://clarkparsia.com/pellet/.

Input: a target concept C, background knowledge $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a training set $E = Ind^{+}_{\mathsf{C}}(\mathcal{A}) \cup Ind^{-}_{\mathsf{C}}(\mathcal{A}) \subseteq Ind(\mathcal{A})$ containing assertions on concept C.
Output: induced concept definition C.

```
Repeat
    Initialize $\mathsf{C}' = \bot$
    Compute hypotheses $\mathsf{C}' \sqsupseteq H_1, \ldots, H_n$ based on refinement operators for $\mathcal{ALC}$
        logic
    Let $h_1, \ldots, h_n$ be features of the probabilistic Noisy-OR classifier, apply the EM
        algorithm
    For all $h_i$
        Compute score $\prod_{e_p \in E_p} covers(e_p, h_i, B)$
    Let $h'$ the hypothesis with the best score
According to $h'$ add $H'$ to $\mathsf{C}$
Until $score(\{h_1, \ldots, h_i\}, \lambda_i, E) > score(\{h_1, \ldots, h_{i+1}\}, \lambda_{i+1}, E)$
```

Fig. 2. Complete learning algorithm.

the EM algorithm of the Noisy-OR classifier to estimate the class ascribed to the instance.

In order to learn probabilities associated to terminologies obtained for the former algorithm we commonly resort to the EM algorithm. In this sense, we are influenced in several respects from approaches given in [16].

4 Preliminary Results

To demonstrate feasibility of our proposal, we have run preliminary tests on relational data extracted from the Lattes curriculum platform, the Brazilian government scientific repository[4]. The Lattes platform is a public source of relational data about scientific research, containing data on several thousand researchers and students. Because the available format is encoded in HTML, we have implemented a semi-automated procedure to extract content. A restricted database has been constructed based on randomly selected documents. We have performed learning of axioms based on elicited asserted concepts and roles, further probabilistic inclusions have been added according to the CR$\mathcal{ALC}$ syntax. Figure 3 illustrates the network generated for a domain of size 2.

For instance, to properly identify a professor, the following concept description has been learned:

$$\begin{aligned}\mathsf{Professor} \equiv\ & \mathsf{Person} \\ & \sqcap(\exists \mathsf{hasPublication.Publication} \sqcup \exists \mathsf{advises.Person} \sqcup \exists \mathsf{worksAt.Organization})\end{aligned}$$

When $\mathsf{Person}(0)$ [5] is given by evidence, the probability value $P(\mathsf{Professor}(0)) = 0.68$ (we have considered a large number of professors in our experiments), as

[4] http://lattes.cnpq.br.
[5] Indexes $0, 1 \ldots n$ represent individuals from a given domain.

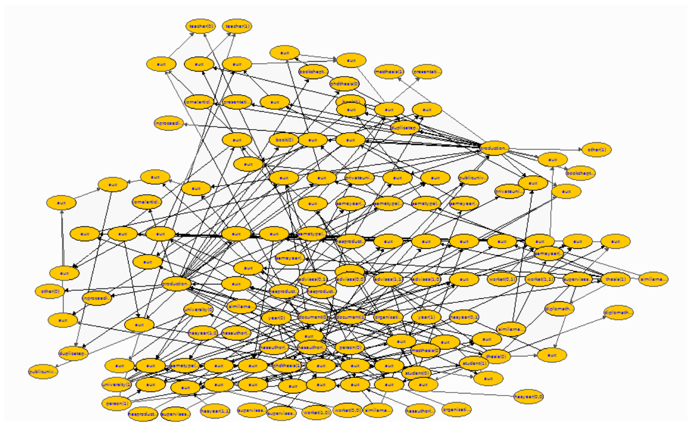

Fig. 3. Relational Bayesian network for the Lattes curriculum dataset.

further evidence is given the probability value changes to:

$$P(\mathsf{Professor}(0)|\exists\mathsf{hasPublication}(1)) = 0.72,$$

and

$$P(\mathsf{Professor}(0)|\exists\mathsf{hasPublication}(1) \sqcup \exists\mathsf{advises}(1)) = 0.75.$$

The former concept definition can conflict with standard ILP approaches, where a more suitable definition might be mostly based on conjuntions. In contrast, in this particular setting, the probabilistic logic approach has a nice and flexible behavior. However, it is worth noting that terminological constructs basically rely on the refinement operator used during learning.

Another query, linked to relational classification, allows us to prevent duplicate publications. One can be interested in retrieving the number of publications for a given research group. Whereas this task might seem trivial, difficulties arise mainly due to multi-authored documents. In principle, each co-author would have a different entry for the same publication in the Lattes platform, and it must be emphasized that each entry is be prone to contain errors. In this sense, a probabilistic concept for duplicate publications was learned:

$$\begin{aligned}\mathsf{DuplicatePublication} \equiv\ & \mathsf{Publication} \\ & \sqcap(\exists\mathsf{hasSimilarTitle.Publication} \sqcup \exists\mathsf{hasSameYear.Publication} \\ & \sqcup\mathsf{hasSameType.Publication}))\end{aligned}$$

It clearly states that a duplicate publication is related to publications that share similar title[6], same year and type (journal article, chapter book and so on). At first, the prior probability is low: $P(\mathsf{DuplicatePublication}(0)) = 0.05$. Evidence on title similarity increases considerably the probability value:

$$P(\mathsf{DuplicatePublication}(0)|\exists\mathsf{hasSimilarTitle}(0,1)) = 0.77.$$

[6] Similarity was carried out by applying a "LIKE" database operator on titles.

Further evidence on type almost guarantees a duplicate concept:

$$P(\mathsf{DuplicatePublication}(0)|\exists\mathsf{hasSimilarName}(1) \sqcap \exists\mathsf{hasSameType}(1)) = 0.99.$$

It must be noted that title similarity does not guarantee a duplicate document. Two documents can share the same title (same author), but nothing prevents them from being published on different means (for instance, a congress paper and an extended journal article). Probabilistic reasoning is valuable to deal with such issues.

5 Conclusion

In this paper we have presented algorithms that perform learning of both probabilities and logical constructs from relational data for the recently proposed Probabilistic DL CR$\mathcal{ALC}$. Learning of parameters is tackled by the EM algorithm whereas structure learning is conducted by a combined approach relying on statistical and ILP methods. We approach learning of concepts as a classification task; a Noisy-OR classifier has been accordingly adapted to do so.

Preliminary results have focused on learning a probabilistic terminology from a real-world domain — the Brazilian scientific repository. Probabilistic logic queries have been posed on the induced model; experiments suggest that our methods are suitable for learning ontologies in the Semantic Web.

Our planned future work is to investigate the scalability of our learning methods.

Acknowledgements

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

References

1. Lisi, F.A., Esposito, F.: Foundations of onto-relational learning. In: ILP '08: Proceedings of the 18th International Conference on Inductive Logic Programming, Berlin, Heidelberg, Springer-Verlag (2008) 158–175
2. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence **26**(2) (2007) 139–159
3. Lehmann, J., Hitzler, P.: A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In Blockeel, H., Shavlik, J.W., Tadepalli, P., eds.: ILP '08: Proceedings of the 17th International Conference on Inductive Logic Programming. Volume 4894 of Lecture Notes in Computer Science., Springer (2008) 147–160
4. Muggleton, S.: Inductive Logic Programming. McGraw-Hill, New York (1992)
5. Cozman, F.G., Polastro, R.B.: Loopy propagation in a probabilistic description logic. In: SUM '08: Proceedings of the 2nd International Conference on Scalable Uncertainty Management, Berlin, Heidelberg, Springer-Verlag (2008) 120–133

6. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
7. de Campos, C.P., Cozman, F.G., Ochoa-Luna, J.E.: Assembling a consistent set of sentences in relational probabilistic logic with stochastic independence. Journal of Applied Logic **7** (2009) 137–154
8. Hailperin, T.: Sentential Probability Logic. Lehigh University Press, Bethlehem, United States (1996)
9. Cozman, F.G., Polastro, R.: Complexity analysis and variational inference for interpretation-based probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence. (2009)
10. Quinlan, J.R., Mostow, J.: Learning logical definitions from relations. In: Machine Learning. (1990) 239–266
11. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo (1998)
12. Vomlel, J.: Noisy-OR classifier: Research articles. Int. J. Intell. Syst. **21**(3) (2006) 381–398
13. Baader, F., Nutt, W.: Basic description logics. In: Description Logic Handbook. Cambridge University Press (2002) 47–100
14. Fanizzi, N., D'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: ILP '08: Proceedings of the 18th International Conference on Inductive Logic Programming, Berlin, Heidelberg, Springer-Verlag (2008) 107–121
15. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
16. Raedt, L.D., Kersting, K.: Probabilistic inductive logic programming. In: Probabilistic ILP - LNAI 4911. Springer-Verlag Berlin (2008) 1–27
17. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19-20** (1994) 629–679
18. Kietz, J.U.: Learnability of description logic programs. In: Inductive Logic Programming, Springer (2002) 117–132
19. Rouveirol, C., Ventos, V.: Towards learning in CARIN-$\mathcal{ALN}$. In: ILP '00: Proceedings of the 10th International Conference on Inductive Logic Programming, London, UK, Springer-Verlag (2000) 191–208
20. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: $\mathcal{AL}$-log: integrating Datalog and description logics. Journal of Intelligent and Cooperative Information Systems **10** (1998) 227–252
21. Rosati, R.: DL+log: Tight integration of description logics and disjunctive datalog. In: KR. (2006) 68–78
22. Cohen, W., Hirsh, H.: Learning the CLASSIC description logic: Theoretical and experimental results. In: (KR94): Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference, Morgan Kaufmann (1994) 121–133
23. Badea, L., Nienhuys-Cheng, S.H.: A refinement operator for description logics. In: ILP '00: Proceedings of the 10th International Conference on Inductive Logic Programming, London, UK, Springer-Verlag (2000) 40–59
24. Lehmann, J.: 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., Springer (2007) 883–898
25. Inuzuka, N., Kamo, M., Ishii, N., Seki, H., Itoh, H.: Top-down induction of logic programs from incomplete samples. In: ILP '96 : 6th International Workshop. Volume 1314 of LNAI., SV (1997) 265–284
26. Landwehr, N., Kersting, K., DeRaedt, L.: Integrating Naïve Bayes and FOIL. J. Mach. Learn. Res. **8** (2007) 481–507