Terminology Learning through Taxonomy Discovery

Raphael Melo
and Kate Revoredo
Postgraduate Information Systems Program
UNIRIO
Rio de Janeiro, Brazil
raphael.thiago, katerevoredo@uniriotec.br

Aline Paes
Department of Computer Science
Institute of Computing
UFF
Rio de Janeiro, Brazil
alinepaes@ic.uff.br

***Abstract*—**

Description Logics based languages have emerged as the standard knowledge representation scheme for ontologies. Typically, an ontology formalizes a number of dependent and related concepts in a domain, encompassed as a terminology. As defining such terminologies manually is a complex, time consuming and error-prone task, there is great interest and even demands for methods that learn terminologies automatically. Learning a terminology in Descriptions Logics concerns to learn several related concepts. This process would greatly benefit of an ideal order to determine which concept should be learned before another concept. Arguably, such an order would yield rich and readable terminologies, as previously, and interrelated concepts formerly learned could be used to induce the description of further concepts. In this work, we contribute with a formal definition of the concept and terminology learning problems and from such definitions we devise an algorithm for finding an ordering through concept taxonomy discovery, that should be followed when learning several related concepts. We show through an experiment that by following the order detected by the algorithm, we are able to afford a more readable terminology than methods that do not conceive an ideal order or do not learn concepts in a dependent way.

I. Introduction

Domain knowledge representation [1] is an important task when considering applications reasoning over a domain. Description logics (DLs) [2] form a family of representation languages, with different expressive power, that are typically decidable fragments of first order logic (FOL). With DL it is possible to represent domain concepts and relations between them. Moreover, due to their computational and expressibility power, they have been widely used in Web Semantic [3] for Ontology representation.

The task of representing a domain knowledge through a DL is usually done manually, which is time consuming and error prone. Therefore, the consideration of mechanisms for automatically learning a DL is relevant and sometimes even demanding. A number of approaches have been proposed in the literature, as [4] and [5], where the concepts are learned stepwise, i.e., a single concept is learned at each time. This makes necessary one execution of the learning algorithm for each concept being learned. Moreover, the order chosen for learning the concepts determines the quality of the final representation and it is not always clear what is the best order to consider. Thus, one problem that may arise with these approaches is terminology illegibility.

In this paper, we propose an approach for finding the best concept ordering for learning a terminology in DL. For defining this order, first the taxonomy of the concepts is automatically discovered. Moreover, we also contribute with a formal definition of the concept and terminology learning problems. Preliminary experiments conducted with our method, combined with DL-Learner [4] to learn descriptions, show the potential of this proposal.

The paper is organized as follows. In Section II, Description Logic and concept learning are reviewed. In Section III, we present the algorithms developed to find an ideal ordering of concepts and an algorithm that follows such an order to learn concepts. Section IV presents the experiment conducted to validate our proposal. Section V concludes the paper.

II. Background Knowledge

In this section, basic definitions on description logic are reviewed and formal definitions on concept and terminology learning problems are presented.

A. Description Logics

The family of knowledge representation languages largely used to formalize ontologies are called Description Logics [2]. They are decidable fragments of First-order logic which try to achieve a balance between expressibility and complexity.

A DL knowledge base ($\mathcal{KB}$) has two components: a *TBox* and a *ABox*. The *TBox* contains intensional knowledge in the form of a terminology. Knowledge is expressed in terms of *individuals*, *concepts*, and *roles*. Thus, the terminology consists of concepts, which denote a set of individuals and roles which denote binary relationships between individuals. 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. Moreover, in this paper, we assume the common assumption made about DL terminologies: (i) only one definition for a concept name and (ii) concept definitions are acyclic. The *ABox* is composed of assertions about the individuals of the domain. An assertion states that an individual belongs to a concept or that a pair of individuals satisfies a role. Attached to a description logic there must be a reasoning mechanism, responsible for inferring information about individuals from ($\mathcal{KB}$). Figure 1 show an example of a $\mathcal{KB}$.

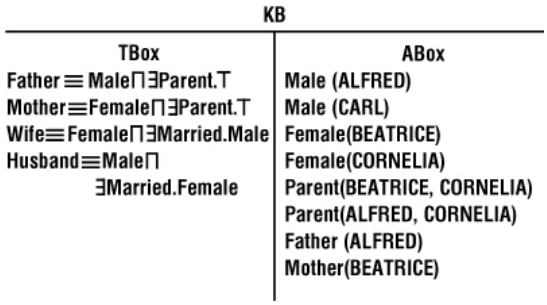

Fig. 1. Example of a Knowledge Base

B. Concept Learning

Modeling a terminology of a domain in DL and subsequent queries within it are essential tasks in computer and information science. However, besides being expensive to manually define a terminology, this is also an error prone task, even more because the domain experts themselves do not always agree about the definitions of concepts and relationships among them [6]. Therefore, in order to effectively exploit such terminologies, especially when dealing with large amounts of data such as in the semantic web [3], it is necessary to apply techniques from machine learning [7] for automatically learning a terminology from data.

Since the concepts are the basic component of a terminology, the first step is to conceive how to learn a concept from data. A number of algorithms have been proposed in the literature to learn concepts in DL [4], [5], [8]. The most prominent and functionally implemented examples of such algorithms are in DL-FOIL [5] and DL-Learner [9] systems. Both are inspired on techniques developed within the field of *Inductive Learning Programming* (ILP) [10], whose general goal is to automatically induce logic programs from data. When learning a concept, one has the purpose of finding a generalized and correct definition of such a concept from a set of examples, as defined below:

Definition 1 (Concept Learning)**.** *Given:*

- *a knowledge base $\mathcal{KB}$,*
- *a target concept $Target$ such as $Target \notin \mathcal{KB}$,*
- *a set of target examples $\mathcal{E}$, divided into positive ($\mathcal{E}_p$) and negative ($\mathcal{E}_n$) examples, such that $\mathcal{E} = \mathcal{E}_p \cup \mathcal{E}_n$*

Find:

- *A definition of the concept $C(Target \equiv C)$ such that $\mathcal{KB} \cup C \models \mathcal{E}_p$ and $\mathcal{KB} \cap C \not\models \mathcal{E}_n$.*

Although Definition 1 requires that the learned concept covers all the positive examples and none of the negative examples, usually this hard criteria is relaxed, to make possible the induction of a concept as good as possible.

An algorithm for learning concepts in description logics is composed of the usual following elements [4] :

1) a refinement operator to build the search tree of concepts;
2) a search algorithm to control how this search tree is traversed;
3) a scoring function to evaluate the nodes of the tree and to point out the best current concept candidate.

The refinement operator is the most impactful component when building a concept definition. The concepts may be built either from a general concept to a more specific one, through downward operators, or the other way around, i.e., from a more specific concept to a more general one, through upward refinement operators [4]. When learning a concept written in description logics, the definitions belonging to the TBox are employed to build new definitions for additional concepts (Example 1).

Example 1. *Let $\mathcal{KB}$ be as presented in Figure 1. The target concept $Target$ is* SINGLEFATHER*. A sound concept definition found could be* SINGLEFATHER $\equiv Father \sqcap \forall Married.\bot$.

Given the concept learning task defined as $A_C =< \mathcal{KB}, \mathcal{E} >$, the formal definition of the terminology learning task is shown in Definition 2.

Definition 2 (Terminology Learning)**.** *Learning a terminology $\mathcal{T}$ means defining the partial order $A_\mathcal{T} =< A_{C_1}, A_{C_2}, \ldots, A_{C_n} >$, composed by n concept learning tasks (A_{C_i}).*

Therefore, learning a complete terminology means defining an order for the elements in $A_\mathcal{T}$. Consider that the definition of a concept C_i learned through A_{C_i} is included in the knowledge base $\mathcal{KB}$. Thus, the definition of a concept C_j, learned through A_{C_j}, where $j > i$, can be based on C_i. Therewith, it is possible to state that the interpretability of the terminology $\mathcal{T}$ is directly related to the partial order $A_\mathcal{T}$ defined. In other words, the order in which the concepts are learned is relevant.

III. Learning DL through Concept Ordering

As previously discussed, the readability of a learned terminology and the complexity of the concepts embedded on it, are closely related to the order that such concepts are learned. Finding out the subsumption relations between concepts will make possible to yield an ordering of the concepts, which in turn will support the learning of a rich – although simple and easily understandable – terminology. The question that arises is how to obtain a subsumption order from concepts that have not been learned yet. We tackle this problem in the paper by developing a procedure capable of finding out the relations among concepts *before* learning the concepts, by taking advantage of their set of examples.

A. Concepts Dependency

When learning a concept through machine learning techniques, the set of examples is a relevant component of the learning process, in the sense that they are responsible for molding the concept definition, by distinguishing one concept from another. Therefore, the set of positive and negative examples provide a strong evidence for the discovery of dependencies and relationships among concepts. Thus, in this paper we propose a pre-learning phase for discovering concept dependency.

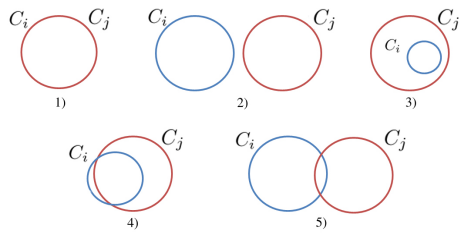

Fig. 2. The relationship among concepts C_i and C_j through their positive examples, represented as Venn diagrams

Let $A_{Ci} =< \mathcal{KB}, \{\mathcal{E}_{p_i}, \mathcal{E}_{ni}\} >$ and $A_{Cj} =< \mathcal{KB}, \{\mathcal{E}_{p_j}, \mathcal{E}_{nj}\} >$, where $\mathcal{E}_{p_i} \subseteq ABox$ and $\mathcal{E}_{p_j} \subseteq ABox$ ($\mathcal{E}_{ni}$ and $\mathcal{E}_{nj}$ are negative assertions, thus $\mathcal{E}_{ni} \nsubseteq ABox$ and $\mathcal{E}_{nj} \nsubseteq ABox$). A comparative analysis of the individuals belonging to $\mathcal{E}_{p_i}$ and $\mathcal{E}_{p_j}$ points out how strongly related the corresponding concepts are and what is the best order ($\preccurlyeq$) for learning them. Example 2 illustrate this relation.

Example 2. *Let $\mathcal{E}_{p_1}$ = {john,bob,edd} and $\mathcal{E}_{p_2}$ = {bob} to be examples for concepts* FATHER *and* GRANDFATHER *respectively. $\mathcal{E}_{p_2} \subseteq \mathcal{E}_{p_1}$, then the definition of concept* FATHER *will include more individuals then the definition of concept* GRANDFATHER*, thus being more general. Therefore, we argue that in the definition of concept* GRANDFATHER *concept* FATHER *can be used, but not the other way around. Thus, if* FATHER *is learned and added to $\mathcal{KB}$ before learning* GRANDFATHER*, then* FATHER *can be used to compose the definition of* GRANDFATHER*. A learning order is then established between the two concepts (* FATHER $\preccurlyeq$ GRANDFATHER*).*

The comparative analysis between two concepts (C_i and C_j) through their sets of positive examples is comprised in the following cases (Figure 2 depicts these cases):

1) Identical sets ($C_i \equiv C_j$): all elements belonging to one set also belong to the other set;
2) Disjoint sets ($C_i \cap C_j = \emptyset$): both sets have no element in common;
3) Inner set ($C_i \subseteq C_j$): one set is a subset of the other;
4) High intersection: the proportion of shared elements is higher than a threshold;
5) Low intersection: the proportion of shared elements is lower than a threshold.

The cases 1, 2 and 3, are the ideal cases because the underlying relationships between concepts are clear; on the other hand, the cases 4 and 5, need some mapping to one of the ideal cases, this is accomplished via a threshold. Therefore, the case 3 is a special case of 4, when the threshold is equal to 1; the case 2 is a special case of 5, when the threshold is equal to 0.

The explicit relations among the concepts can be translated to a subsumption tree (*a taxonomy*). In this way, the relations among all concept definitions may be obtained from the comparative analysis of each pair of concepts that belong to the terminology learning task $A_{\mathcal{T}}$. Such an analysis is the main component for building a tree representing the relations among the concepts. This tree, which is mapping a taxonomy among the concept definitions, can be used to determine the best ordering to learn the concept definitions.

B. Learning Concept Ordering

In Section III-A, five possible results were detected after analyzing the relations between two concepts (C_i and C_j) through their sets of positive examples. From the first three cases, it is possible to directly acquire the relationship of the concepts. However, since the two remaining cases contemplate types of intersection between C_i and C_j, a further analysis is required aiming to decide whether a subsumption relation will be considered or a total independence between C_i and C_j will be a better choice. In this section, a procedure concerning the five possibilities is devised, which is going to lead to a concept taxonomy.

The Algorithm 1 presents our procedure for learning a concept taxonomy that will be later used to define an order for learning the concepts of a terminology.

Algorithm 1 Algorithm for learning a concept taxonomy

Require: a terminology learning task ($A_{\mathcal{T}} =< A_{C_1}, A_{C_2}, \ldots, A_{C_n} >$, where all $A_{C_i} =< \mathcal{KB}, \mathcal{E}_i >$ is a concept learning task for concept C_i
Ensure: a concept taxonomy $\mathcal{T}$

1: Starts $\mathcal{T}$ with a default concept in the root;
2: Add each concept C_i considered in $A_{\mathcal{T}}$ as a child of the root;
3: **for** each A_{Ci}, $A_{Cj} \in A_{\mathcal{T}}$ **do**
4: $\quad \mathcal{T} \leftarrow$ call Algorithm 2 with A_{Ci}, A_{Cj}, $\mathcal{T}$
5: **end for**
6: **return** $\mathcal{T}$

The algorithm receives as input a terminology learning task, as outlined in Definition 2, and returns a taxonomy of its concepts.

Given the ordering tree, a concept can only be defined if all its parents have already been defined. The root of the tree is a default concept which must be the father of all the concepts that do depend on any other concept in $A_{\mathcal{T}}$.

The algorithm 1 starts by creating a tree with only the root. Next, all concepts (C_i) considered in a concept learning task (A_{Ci}), where $A_{Ci} \in A_{\mathcal{T}}$, are included in the tree as children of the root. Then, Algorithm 2 is called for all pair of concepts C_i and C_j in order to find their relationship. After checking all dependencies, the final concept taxonomy is returned.

In Algorithm 2 the relationship between the positive examples ($\mathcal{E}_p$) sets of two concepts are analyzed in order to resolve in which of the five cases discussed in Section III-A it belongs. The algorithm begins by creating a variable that will hold the amount of shared individuals between the example sets (line 1). The first case checked (line 2) is the one with two equivalent concepts, if that's the case the taxonomy is not modified. That was a choice made in order to maintain both concepts in the resulting terminology. Then it's verified if the concepts are disjoint (line 6), if so there is no relationship

between the concepts, thus the taxonomy will not be changed. If one concept's set is a subset of the other (line 9, 13) then a relationship between them must exist on the resulting taxonomy, the *subsumee* concept is put as a child of the *subsumer* in the taxonomy. Moreover, the relationship between the *subsumee* and the root is no longer necessary. There are two remaining cases, a high intersection between concepts or a low one. The difference is defined using a threshold that needs to be set before starting the process. The amount of intersection can be calculated by finding the smaller concept (relating to the size of the examples set) and dividing the number of shared individuals and its examples set size (line 19). If this value is higher than the threshold, a relationship exists, so the larger concept is added as a *subsumer* in the taxonomy and the smaller concept as its *subsumee* (line 20); again the relationship between root and the *subsumee* can be removed. The important thing to notice is that the only parent removed from the *subsumee* is the root, all the other parents it may have are maintained, so the right order can be obtained.

Algorithm 2 Algorithm for Ascertaining Dependencies among Concepts in a Terminology

Require: Concept learning tasks $A_{Ci} =< \mathcal{KB}, \{\mathcal{E}_{p_i}, \mathcal{E}_{ni}\} >$ and $A_{Cj} =< \mathcal{KB}, \{\mathcal{E}_{p_j}, \mathcal{E}_{nj}\} >$; a ordering tree $\mathcal{T}$.
Ensure: An updated ordering tree $\mathcal{T}$

1: sharedIndividuals $\leftarrow 0$
2: **if** $A_{Ci} \equiv A_{Cj}$ **then**
3: **return** $\mathcal{T}$
4: **end if**
5: sharedIndividuals $\leftarrow |\mathcal{E}_{p_i} \cap \mathcal{E}_{p_j}|$
6: **if** sharedIndividuals = 0 **then**
7: **return** $\mathcal{T}$
8: **end if**
9: **if** $\mathcal{E}_{p_i} \subseteq \mathcal{E}_{p_j}$ **then**
10: include A_{Cj} as father of A_{Ci} in $\mathcal{T}$
11: remove ROOT from A_{Ci} parents set
12: **end if**
13: **if** $\mathcal{E}_{p_j} \subseteq \mathcal{E}_{p_i}$ **then**
14: include A_{Ci} as father of A_{Cj} in $\mathcal{T}$
15: remove ROOT from A_{Cj} parents set
16: **end if**
17: **if** $|\mathcal{E}_{p_i}| \leq |\mathcal{E}_{p_j}|$ **then**
18: smaller_concept $\leftarrow A_{Ci}$
19: larger_concept $\leftarrow A_{Cj}$
20: **else**
21: smaller_concept $\leftarrow A_{Cj}$
22: larger_concept $\leftarrow A_{Ci}$
23: **end if**
24: **if** (sharedIndividuals / |smaller_concept| $\geq$ threshold) **then**
25: include larger_concept as father of smaller_concept in $\mathcal{T}$
26: remove ROOT from smaller_concept parents set
27: **end if**
28: **return** $\mathcal{T}$

C. Learning Terminology through concept ordering

Algorithm 3 presents the overall procedure for learning a terminology. It requires as input the terminology learning task

Algorithm 3 Top-Level Algorithm for Learning Terminologies

Require: A terminology learning task $A_{\mathcal{T}}$.
Ensure: A Terminology $\mathcal{TE}$

1: find a Taxonomic Tree $\mathcal{T}$ through Algorithm 1
2: find an Ordered Sequence $A_{\mathcal{S}}$ of concept learning tasks $A_{\mathcal{T}}$ using $\mathcal{T}$, where concepts only come after their parents
3: **for** each $A_{Ck} \in A_{\mathcal{S}}$, in the established order **do**
4: learn a description $\mathcal{D}$ for C_k from A_{Ck}, using any algorithm that learns concepts in Description Logics
5: include $\mathcal{D}$ in $\mathcal{KB}$ for all $\mathcal{A}_{\mathcal{C}i} \in A_{\mathcal{S}}$ where $i > k$
6: **end for**
7: **return** $\mathcal{TE}$

$A_{\mathcal{T}}$. As in our method it is necessary to find a taxonomy tree, so that a concept ordering is induced from such a tree, the first step of the algorithm is to call the procedure devised as Algorithm 1, which is going to return the tree comprising each concept addressed by the terminology problem.

Next, it is necessary to establish the order that the algorithm is going to follow to learn each concept description. This is done by traversing the tree in a breadth-first manner, and collecting all nodes at a level in order. In this way, it is possible to guarantee that a concept is only tackled after the descriptions for all its parents have been learned.

Finally, each concept is learned gradually in order, through a loop that visits the concept learning task in the order defined in the second main step of the algorithm. To learn a description, the concept itself, its positive and negative examples and the current TBox is submitted to a description logic learning algorithm, such as DL-Learner [4], DL-FOIL [5] or Yin-Yang [8]. Then, the description found is included in the TBox of all $\mathcal{KB}$ of the following $\mathcal{A}_{\mathcal{C}i}$, so that it can be used in next iterations of the loop, as part of other descriptions. In this way, we achieve the learning of related concepts as found out in Algorithm 2, in a rather ideal order discovered from the tree learned by the Algorithm 1.

The algorithm ends by returning the terminologies learned at each step of the loop.

IV. Experiment

In order to validate the proposed method for terminology learning task, we used the algorithm proposed (3) on a knowledge base supported by the kinship concepts [11], which adresses the relationships within a family setting. Five concepts were chosen from within the kinship concepts to become the concepts to be learned, as follows:

- GRANDPARENT(GP)
- GREATGRANDPARENT(GG)
- BROTHER(BR)
- UNCLE(UC)
- NEPHEW(NP)

Those concepts were chosen because they cover all the cases discussed in Section III-A, except for the equivalence

TABLE I. PERCENTAGE OF SHARED INDIVIDUALS RETURNED BY ALGORITHM 2

	GP	GG	BR	UC	NP
GP		**1.00**	0.19	0.43	0.04
GG	0.33		0.03	0.07	0.00
BR	0.25	0.13		**1.00**	**0.92**
UC	0.25	0.13	0.45		0.38
NP	0.04	0.00	0.71	0.64	

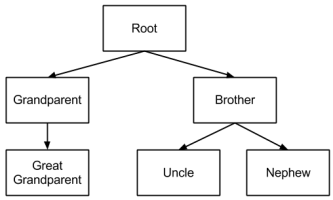

Fig. 3. Taxonomy found By Algorithm 1

one, making this set of concepts acceptable and simple to validate the method proposed.

The knowledge base used in the experiments is the one provided in http://www.cs.utexas.edu/users/ml/forte.html. The ABox has assertions about 86 individuals, encompassing their genres and sibling, parenthood and matrimonial relations. Those assertions were used to produce the sets of examples for each of the five concept learning problems. Each individual cited in the knowledge base is either a positive example or a negative example to a concept.

a) Comparison Methodology: In order to validate the contribution of our approach, the following experiments were conducted for purposes of comparison:

1) Learning concepts *individually*: Concepts are learned independently without considering a determined order and no concept description is learned before it is added to the knowledge base;
2) Learning concepts following a *random order*: A random order was generated ($<$ UNCLE, NEPHEW, GREATGRANDPARENT, GRANDPARENT, BROTHER $>$), to simulate the situation where no algorithm is used to compute a good concept ordering. In this case, after each learning problem the learned description is added to the knowledge base.
3) Learning concepts following a learned ordering: this experiment takes into account the algorithms proposed in this paper: Terminologies are learned with Algorithm 3, which in its turn call Algorithm 1 to learn a taxonomy tree.

The description learning component chosen to learn the concepts is the DL-Learner system with default settings, since it is a largely used environment to learn DLs.

b) Results: Table I shows the results obtained within algorithm 2 to discover concept dependency. The value in a cell is the number of shared examples over the size of $\mathcal{E}_p$ for the concept in the column. We arbitrarily considered the threshold to indicate a high intersection case (see section III-A) as 0.8, further experiments on the impact of this value are needed. In these conditions, three relationships were found by the algorithm: i) GRANDPARENT $\preccurlyeq$ GREATGRANDPARENT; ii) BROTHER $\preccurlyeq$ UNCLE; iii) BROTHER $\preccurlyeq$ NEPHEW. Figure 3 shows the taxonomy tree returned by Algorithm 1.

Table II exhibits the concepts learned by each method described in IV-0a. As expected, the descriptions found by our proposed method presents a greater readability, compared to the others two methods. Notice, for example, that the description of GREATGRANDPARENT concept was built from GRANDPARENT concept, which is rather natural in this domain.

Furthermore, the concept UNCLE is found in the description of NEPHEW concept. Although the taxonomy discovery algorithm have included both of them in the same level in the tree, UNCLE description is learned before than NEPHEW description. Algorithm 3 considers any concept previously learned as a possible candidate to compose a description. It is the task of the learning description algorithm to tackle cases like that. To avoid such a situation, it may be necessary to learn a full order for the concepts that happen to be in the same level in the tree. Additionally, this can be mitigated if all the same level learning problems are executed concurrently, which is likely to achieve the full potential of our method. We intend to investigate these questions in future work.

Observing the concepts learned following a random ordering, we can see that the learning terminology algorithm obtained an unusual description to the concept GREATGRANDPARENT: it takes into account the concept UNCLE, which are somewhat unrelated concepts. Moreover, when comparing with the results of the "learning individually" experiment some of the definitions found were semantically incorrect, even though this only happened because of this ABox particular configuration.

To sum up, these results point out the potential of the proposed algorithms, as the descriptions learned through them are readily understood. The random order experiment, on the other hand, generated worse results than the one where the concepts were learned individually, showing that it is possible to obtain worse results when considering a poor order than when formerly learned concepts are not employed at all to learn later concepts.

V. CONCLUSIONS

In this paper, we presented a formal definition of concept and terminology learning tasks, addressing their differences and relationships. Moreover, we developed a method for terminology learning, based on finding an ideal ordering for concept learning tasks, according to a measure of intersection in the set of examples. The order returned by our method is represented by a taxonomy tree which in turn is mapped to a sequential list, defining which concepts should be learned before each other. In this way, each time a description is learned it can take into account the formerly induced concepts that are related to it. Thus, the method has the ability to find a natural and easily readable terminology.

We validated our method by tackling a terminology based on the kinship domain. We compared the algorithms proposed

TABLE II. RESULTING CONCEPTS DEFINITION ON ALL LEARNING EXPERIMENTS

Experiment	Concept	Definition
Concept Learning	GP	EXISTS parent.EXISTS parent.TOP.
	GG	EXISTS parent.EXISTS parent.EXISTS parent.TOP.
	BR	(male AND EXISTS sibling.TOP).
	UC	(male AND EXISTS sibling.EXISTS parent.TOP).
	NP	(male AND $\leq$ 1 sibling.ALL sibling.EXISTS parent.female AND (EXISTS sibling.$\leq$ 1 parent.male OR $\leq$ 0 married.TOP)).
Random Order	GP	EXISTS parent.EXISTS parent.TOP.
	GG	EXISTS parent.(uncle AND EXISTS parent.EXISTS parent.TOP).
	BR	(male AND EXISTS sibling.TOP).
	UC	(male AND EXISTS sibling.EXISTS parent.TOP).
	NP	(male AND $\leq$ 1 sibling.EXISTS parent.uncle AND ($\geq$ 3 sibling.TOP OR $\leq$ 0 married.EXISTS parent.uncle)).
Proposed Method	GP	EXISTS parent.EXISTS parent.TOP.
	GG	(grandparent AND EXISTS parent.grandparent).
	BR	(male AND EXISTS sibling.TOP).
	UC	(male AND EXISTS sibling.EXISTS parent.TOP).
	NP	(male AND $\leq$ 1 sibling.grandparent AND (EXISTS sibling.male OR $\leq$ 0 EXISTS.parent.uncle).

in this paper to (1) learning each concept independently from each other, therefore, with no established order and (2) learning concepts considering a random order, which is likely to be not an ideal ordering for the concepts. From the learned descriptions we can conclude that our method induces clearer and more compact descriptions than the others, by taking advantage of the descriptions previously learned in a instinctive order. We also observed that when the concepts are learned independently from each other, without considering the previously induced concepts, the final descriptions used to be less complex than when considering a random order for learning each description. From such results, it was possible to show the potential of our method for learning terminologies. Experiments on different datasets are needed to eliminate possible biases.

We believe that the proposed approach would bring significant benefits when learning *probabilistic description logics* [12], since the probability values founded are dependent on the pre existing values, therefore defining it in a better order may yield better results.

To the best of our knowledge, the literature related to terminology learning considering the induction of a set of related but distinct concepts is rather scarce. Most work induce terminologies either independently or using an ad-hoc order [13]. However, there is an interesting work related to ours that conceives the analysis of formal concepts to find out the best way to learn a terminology [14]. We intend to further explore this work in future contributions.

ACKNOWLEDGMENTS

The first author would like to thank CAPES for the financial support through a master scholarship.

REFERENCES

[1] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*, 1st ed., M. Kaufmann, Ed. Elsevier, 2004.

[2] F. Baader and W. Nutt, "Basic description logics," in *The description logic handbook*, 2nd ed., F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, may 2010, pp. 47–100.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 5, no. 284, p. 34, 2001.

[4] J. Lehmann and P. Hitzler, "Concept learning in description logics using refinement operators," *Machine Learning*, vol. 78, no. 1-2, pp. 203–250, 2010.

[5] N. Fanizzi, C. d'Amato, and F. Esposito, "Dl-foil concept learning in description logics," in *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP-2008)*, ser. Lecture Notes in Computer Science, vol. 5194 LNAI. Springer, 2008, pp. 107–121.

[6] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems and Their Applications*, vol. 16, no. 2, pp. 72–79, 2001.

[7] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[8] L. Iannone, I. Palmisano, and N. Fanizzi, "An algorithm based on counterfactuals for concept learning in the semantic web," *Applied Intelligence*, vol. 26, no. 2, pp. 139–159, 2007.

[9] J. Lehmann, "Dl-learner: Learning concepts in description logics," *Journal of Machine Learning Research*, vol. 10, pp. 2639–2642, 2009.

[10] L. De Raedt, *Logical and Relational Learning*. Springer, 2008.

[11] L. H. Morgan, *Systems of consanguinity and affinity of the human family*. Smithsonian Institution,, 1870.

[12] K. Revoredo, J. Ochoa-Luna, and F. Cozman, "Learning probabilistic description logics: A framework and algorithms," in *In proceedings of the MICAI*, ser. LNCS, vol. 7094. Springer, 2011, pp. 28–39.

[13] F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro, "Knowledge-intensive induction of terminologies from metadata," in *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 3298. Springer, 2004, pp. 441–455.

[14] F. Distel, "Learning description logic knowledge bases from data using methods from formal concept analysis," Ph.D. dissertation, TU Dresden, 2011.