Induction of Concepts in Web Ontologies through Terminological Decision Trees

Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito

Dipartimento di Informatica
Università degli studi di Bari "Aldo Moro"
Campus Universitario – Via Orabona 4, 70125 Bari, Italy
{fanizzi,claudia.damato,esposito}@di.uniba.it

Abstract. A new framework for the induction of logical decision trees is presented. Differently from the original setting, tests at the tree nodes are expressed with Description Logic concepts. This has a number of advantages: expressive terminological languages are endowed with full negation, thus allowing for a more natural division of the individuals at each test node; these logics support the standard ontology languages for representing knowledge bases in the Semantic Web. A top-down method for inducing *terminological decision trees* is proposed as an adaptation of well-known tree-induction methods. This offers an alternative way for learning in Description logics as concept descriptions can be associated to the terminological trees. A new version of the System TermiTIS, implementing the methods, is experimentally evaluated on ontologies from popular repositories.

1 Introduction

Among the other facets, the Semantic Web (SW) is a Web of Data[1]. Countless structured data sets are distributed all over the Web and containing all kinds of information. They are the property of companies or institutions that tend to make them broadly accessible. Next generation knowledge bases are envisioned to be based on shared ontologies and a number of distributed repositories that contain resources which are annotated using the concepts and properties defined in terms of such ontologies expressed through such standardized representations.

The collection of SW technologies (RDF/S, OWL, etc.) provides an environment where applications can query that data, draw inferences using vocabularies, etc. They are ultimately based on Description Logics (DLs). These languages constitute particular fragments of Predicate Logic (with extensions to higher order logics for the most expressive ones). They differ from the clausal representations for having a variable-free syntax and especially the *open-world semantics* [1] which makes them particularly well suited for Web-scale distributed scenarios.

These considerations justify the growing interest in investigating machine learning and knowledge discovery methods that cope with these formalisms.

[1] See http://www.w3.org/standards/semanticweb/data

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part I, LNAI 6321, pp. 442–457, 2010.

Early works on learning in DLs essentially focused on the learnability of terminological languages, like CLASSIC and its successors [2], that are the ancestors of the current DL languages. More recently, DL concept learning approaches based on refinement operators have been investigated [3]. In [4] a coverage algorithm is presented whose downward operator exploits the notion of *counterfactuals*. This has been exploited also for other related tasks, such as conceptual clustering [5]. Since the algorithm works on examples expressed as *most specific concepts* (MSCs) [1], it tends to provide correct yet excessively complex concepts definitions. To avoid this problem, other top-down refinement algorithm, based on new downward operators, utilizes a syntactic heuristic to guide the search towards correct definitions of limited complexity [6]. A similar approach is followed in DL-FOIL [7], which adapts the well-known learning method to the different representation.

The induction of decision trees is among the most well-known machine learning techniques [8], also in its more recent extensions that are able to work with more expressive logical representations in clausal form [9]. In this work, the general framework is extended to to cope with yet more different logical representations as those designed for formal Web ontologies. We adopt an expressive DL language for representing the tests at the tree nodes of a logical decision tree. This allows to express different concepts w.r.t. the original clausal representation and is naturally compliant with the open-world semantics which is required by the evolving distributed environments for SW applications.

The tree-induction algorithm adopts a classical top-down *divide-and-conquer* strategy [10] which differs from previous DL concept learning methods based on sequential covering or heuristic search, with the use of refinement operators for DL concept descriptions [4, 7, 6]. Once a terminological tree is induced, similarly to the logical decision trees, a definition of the target concepts can be drawn exploiting the nodes in the tree structure. The algorithm has also a useful side-effect: the suggestion of new intermediate concepts which may have no definition in the current ontology. From a technical viewpoint, the likely chance that an instance a might not be assigned[2] to a given concept C (nor to its complement) requires a different setting for the learning problem [7], that is similar to learning with unknown class attributes [11], with a special treatment of the cases of uncertain classification mentioned above.

The resulting system, TERMITIS (TERMI*nological* T*ree* I*nduction* S*ystem*), ver. 1.2, was applied, for comparative purposes, to ontologies that have been considered in previous experiments with other DL learning systems [4, 7]. As they are real ontologies artificial learning problems were crafted by randomly building concept descriptions and determining the respective training and test sets. This also demonstrates the usage of the method as a means for performing approximations of concepts across ontologies (even when they are described with different languages [1]). Standard performance indices require that the class-membership can be determined for each test instance. This is not possible when the open-world semantics is assumed (as discussed before), then we resort to

[2] For the open-world semantics, $\neg C(a)$ does not necessarily follow from $\not\vdash C(a)$.

different indices, measuring the alignment of the classification decided by the inductive model with the one found deductively by a DL reasoner [7]. This allows measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their complements), which may constitute a real added value brought by inductive methods to DL reasoning, although this sort of *abductive* conclusions would have to be validated by a domain expert.

The paper is organized as follows. After the next section introducing the representation, in Sect. 3 the DL learning problem is formalized and discussed. Sect. 4, presents the terminological tree model and the algorithms for growing them and for deriving concept descriptions. In Sect. 5 the experiments proving the effectiveness of the approach are reported. Finally, possible developments are discussed in Sect. 6.

2 Description Logics: Syntax and Semantics

In this section we shortly recall syntax and semantics of the DL representation. For brevity, we cannot report syntax and semantics of the various constructors, which can be easily be found in the reference manual [1]. In turn, the DL concept descriptions are straightforwardly mapped onto XML serializations of the standard ontology languages.

Roughly, the terminological formalisms are concept-centric: they distinguish *concepts* from *relations* that are used to describe restrictions on concepts. In a DL language, primitive *concepts* $N_C = \{C, D, \ldots\}$ are interpreted as subsets of a domain of objects (resources) and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain (properties). *Individuals* represent the objects through names chosen from the set $N_I = \{a, b, \ldots\}$. Complex concept descriptions are built using atomic concepts and primitive roles by means of specific constructors. The meaning of the descriptions is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of the interpretation and the functor $\cdot^{\mathcal{I}}$ stands for the *interpretation function*, mapping the intension of concepts and roles to their extension (respectively, a subset of the domain and a relation defined on such domain).

The *top* concept $\top$ is interpreted as the whole domain $\Delta^{\mathcal{I}}$, while the *bottom* concept $\bot$ corresponds to $\emptyset$. Complex descriptions can be built in $\mathcal{ALC}$ using the following constructors[3]. The language supports *full negation*: given any concept description C, denoted $\neg C$, it amounts to $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. The *conjunction* of concepts, denoted with $C_1 \sqcap C_2$, yields an extension $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ and, dually, concept *disjunction*, denoted with $C_1 \sqcup C_2$, yields $C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$. Finally, there are two restrictions on roles: the *existential restriction*, denoted with $\exists R.C$, and interpreted as the set $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ and the *value restriction*, denoted with $\forall R.C$, whose extension is $\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$.

[3] In fact, $\mathcal{ALC}$ corresponds to the fragment of first-order logic obtained by restricting the syntax to formulae containing two variables. $\mathcal{ALC}$ has a modal logic counterpart, namely the multi-modal version of the logic $\mathbf{K_m}$ [1].

Further constructors extend the expressiveness of the $\mathcal{ALC}$ language. We are interested in the logic that constitutes the counterpart of OWL-DL ontology language, namely $\mathcal{SHOIQ}(D)$, that extends $\mathcal{ALC}$ with transitive roles, role hierarchies, individual classes, inverse roles and qualified number restrictions. Besides, concrete domains[4] (**D**) with their specific semantics can be dealt with.

The set-theoretic notion of *subsumption* between concepts (or roles) can be given in terms of the interpretations:

Definition 2.1 (subsumption). *Given two concept descriptions C and D, C* is subsumed *by D, denoted by $C \sqsubseteq D$, iff for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Hence, $C \equiv D$ amounts to $C \sqsubseteq D$ and $D \sqsubseteq C$.*

This notion may be easily extended to the case of role descriptions (for languages admitting role-hierarchies).

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. $\mathcal{T}$ is a set of terminological axioms $C \sqsubseteq D$, yet we will consider only definitions $A \equiv D$, where $A \in N_C$ is a concept name (atomic) and D is a concept description given in terms of the language constructors, meaning $A^{\mathcal{I}} = D^{\mathcal{I}}$. The ABox $\mathcal{A}$ contains extensional assertions (ground facts) on concepts and roles, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Note that the *unique names assumption* is not necessarily made by default[5].

An interpretation satisfying all the axioms in the knowledge base is said to be a *model* for it. Hence, the usual notions of satisfiability, consistency, etc. apply also for these logics. reasoning is generally performed by resorting to tableau algorithms [1].

Example 2.1 (kinship). Given primitive concepts like `Male` and primitive roles like `hasChild`, an example of TBox (in the proposed language) is:

$$\begin{aligned} \mathcal{T} = \{\ & \texttt{Female} \equiv \neg\texttt{Male}, \\ & \texttt{Father} \equiv \texttt{Male} \sqcap \exists\texttt{hasChild}.\top, \\ & \texttt{Mother} \equiv \texttt{Female} \sqcap \exists\texttt{hasChild}.\top, \\ & \texttt{Parent} \equiv \texttt{Mother} \sqcup \texttt{Father}\ \} \end{aligned}$$

The concept `Father` translates the sentence: *"a father is a male individual that has some individual as his child"* ($\top$ denotes the most general concept). It is easy to see that $\texttt{Father} \sqsubseteq \texttt{Parent}$, yet $\texttt{Parent} \not\sqsubseteq \texttt{Father}$.

Now, if we define a new concept:
$\texttt{FatherWithoutSons} \equiv \texttt{Father} \sqcap \forall\texttt{hasChild}.(\neg\texttt{Male})$
then $\texttt{FatherWithoutSons} \sqsubseteq \texttt{Father}$ yet $\texttt{Father} \not\sqsubseteq \texttt{FatherWithoutSons}$.

ABox assertions are ground facts like:
$\texttt{Father}(\texttt{OEDIPUS})$, $\texttt{Male}(\texttt{OEDIPUS})$, $\texttt{hasChild}.\texttt{Male}(\texttt{JOCASTA},\texttt{OEDIPUS})$,
$\exists\texttt{hasChild}.\texttt{Female}(\texttt{OEDIPUS})$, $\neg\forall\texttt{hasChild}.\texttt{Male}(\texttt{JOCASTA})$ and so on. □

[4] Concrete domains include basic data types, such as numerical types, strings, etc., but also more complex types, such as tuples of the relational calculus, spatial regions, or time intervals.

[5] Different individual names (that ultimately correspond to URIs in RDF/OWL) may be mapped onto the same object (resource), if not explicitly forbidden.

The most important inference service from the inductive point of view is *instance checking* [1], that amounts to ascertain class-membership assertions: $\mathcal{K} \models C(a)$, 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 $\mathcal{K}$. An important difference with other FOL fragments is the *open-world assumption* (OWA) which makes it more difficult to answer class-membership queries. Thus it may happen that an object that cannot be proved to belong to a certain concept is not necessarily a counterexample for that concept. That would only be interpreted[6] as a case of insufficient (incomplete) knowledge for that assertion.

Example 2.2 (Oedipus' family). Given a TBox $\mathcal{T}$ containing the kinship concept definitions of the previous example and also the concept
$\texttt{MotherWithNoDaughter} \equiv \texttt{Mother} \sqcap \forall \texttt{hasChild}.\neg\texttt{Female}$
and the following ABox (using another atomic concept Parricide):

$$\begin{aligned}\mathcal{A} = \{\ & \texttt{Female}(\texttt{JOCASTA}), \texttt{Female}(\texttt{POLYNEIKES}),\\ & \texttt{Male}(\texttt{OEDIPUS}), \texttt{Male}(\texttt{THERSANDROS}),\\ & \texttt{hasChild}(\texttt{JOCASTA}, \texttt{OEDIPUS}),\\ & \texttt{hasChild}(\texttt{JOCASTA}, \texttt{POLYNEIKES}),\\ & \texttt{hasChild}(\texttt{OEDIPUS}, \texttt{POLYNEIKES}),\\ & \texttt{hasChild}(\texttt{POLYNEIKES}, \texttt{THERSANDROS}),\\ & \texttt{Parricide}(\texttt{OEDIPUS}), \neg\texttt{Parricide}(\texttt{THERSANDROS})\}\end{aligned}$$

one may infer the truth for assertions such as Parent(OEDIPUS), but not for MotherWithNoDaughter(POLYNEIKES) because it may well be that a daughter of POLYNEIKES is merely not known.

In order to better appreciate the difference of ABox reasoning w.r.t. query answering let us consider the classic reasoning problem [1]:
given the query (hasChild.(Parricide ⊓ hasChild.¬Parricide))(JOCASTA)? (*does Jocasta have a child that is a parricide and that, in turn, has a child that is not a parricide ?*), a query answering system under a closed-world semantics[7] would return a negative answer (false) because it cannot prove the assertion Parricide(POLYNEIKES). Conversely, by reasoning on the possible models of the ABox, one may divide these interpretations into two classes: one contains those satisfying Parricide(POLYNEIKES) and the other with the models of its negation. In both cases JOCASTA can be recognized as an instance of the query concept, hence the answer to the query is true. □

This assumption is perfectly compatible with the typical scenario related to the Semantic Web, where new resources may continuously be made available across the Web, hence a complete knowledge cannot be assumed at any time.

Another useful inference service provided by the DL reasoners is concept *retrieval*: given a certain concept C, retrieve all the individuals that belong to it. Formally: $\{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models C(a)\}$.

[6] Models could be constructed for both the membership and non-membership case.
[7] Query answering on databases amounts to *finite model checking* [1] (i.e. evaluation of a formula in a fixed finite model).

3 Learning Problems in Description Logics

For the basic terminology on DL languages and terminologies see [1]. Given a DL knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ as the source for the background knowledge, suppose that an expert provides proper ABox assertions to deem some individuals as examples (or counter-examples) of some new target concept for which one wants to learn a definition in the form of a DL concept description:

Definition 3.1 (learning problem). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base.* ***Given***

- *a (new) target concept name C;*
- *a set of positive and negative examples for C:*
  $\mathcal{S}_C^+(\mathcal{A}) = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models C(a)\}$ *and*
  $\mathcal{S}_C^-(\mathcal{A}) = \{b \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models \neg C(b)\}$

Build *a concept description D so that the following relations are satisfied by the definition $C \equiv D$:*

- $\mathcal{K} \models C(a) \quad \forall a \in \mathcal{S}_C^+(\mathcal{A})$ *and*
- $\mathcal{K} \not\models C(b) \quad \forall b \in \mathcal{S}_C^-(\mathcal{A})$

Note that in this supervised concept learning task $\mathcal{S}_C^+(\mathcal{A})$ and $\mathcal{S}_C^-(\mathcal{A})$ (i.e. sets of individuals) represent, resp., the sets of positive and negative examples, whereas D is the hypothesis to be induced. In the original setting for logic decision trees [9] multiple *disjoint* concepts are to be learned. In the case of DLs disjointness must be explicitly stated as an axiom in the knowledge base. Note that this is different from settings where negative examples are such that $\mathcal{K} \not\models C(a)$, that may be fulfilled by a single interpretation satisfying $\neg C(a)$.

The example on the domain of *cars* employed in [9] to illustrate the task of learning logical decision trees can be transposed as follows:

Example 3.1 (car checking). An engineer must check a set of cars. Each car is made of several parts that may need be replaced. This can be done by either their manufacturer or by the engineer himself. In the former case the car is an instance of the `SendBack` concept, in the latter it belongs to the `Fix` concept. In case no worn parts are detected, the car is `Ok`. The TBox $\mathcal{T}$ includes the following background knowledge:

$$\begin{aligned} \{\ \mathtt{Gear} &\sqsubseteq \mathtt{Replaceable},\\ \mathtt{Chain} &\sqsubseteq \mathtt{Replaceable},\\ \mathtt{Engine} &\sqsubseteq \neg\mathtt{Replaceable},\\ \mathtt{Wheel} &\sqsubseteq \neg\mathtt{Replaceable}\ \} \subseteq \mathcal{T} \end{aligned}$$

Besides, since the target concepts are meant to be disjoint, the following axioms must be added to $\mathcal{T}$:

$$\begin{aligned} \{\mathtt{SendBack} &\sqsubseteq \neg(\mathtt{Fix} \sqcup \mathtt{Ok}),\\ \mathtt{Fix} &\sqsubseteq \neg(\mathtt{Ok} \sqcup \mathtt{SendBack}),\\ \mathtt{Ok} &\sqsubseteq \neg(\mathtt{SendBack} \sqcup \mathtt{Fix})\} \end{aligned}$$

The original examples can be encoded as the following set of assertions:

$$\begin{aligned}\mathcal{A}' = \{\ & \mathtt{Machine}(\mathtt{M}_1), \mathtt{hasPart}(\mathtt{M}_1, \mathtt{G}_1), \mathtt{Gear}(\mathtt{G}_1), \mathtt{Worn}(\mathtt{G}_1),\\ & \mathtt{hasPart}(\mathtt{M}_1, \mathtt{C}_1), \mathtt{Chain}(\mathtt{C}_1), \mathtt{Worn}(\mathtt{C}_1),\\ & \mathtt{Machine}(\mathtt{M}_2), \mathtt{hasPart}(\mathtt{M}_2, \mathtt{E}_2), \mathtt{Engine}(\mathtt{E}_2), \mathtt{Worn}(\mathtt{E}_2),\\ & \mathtt{hasPart}(\mathtt{M}_2, \mathtt{C}_2), \mathtt{Chain}(\mathtt{C}_2), \mathtt{Worn}(\mathtt{C}_2),\\ & \mathtt{Machine}(\mathtt{M}_3), \mathtt{hasPart}(\mathtt{M}_3, \mathtt{W}_2), \mathtt{Wheel}(\mathtt{W}_3), \mathtt{Worn}(\mathtt{W}_3),\\ & \mathtt{Machine}(\mathtt{M}_4)\ \} \subseteq \mathcal{A}\end{aligned}$$

Then given this knowledge base and the example sets $\mathcal{S}_C^+(\mathcal{A}) = \{\mathtt{M}_1, \mathtt{M}_3\}$ and $\mathcal{S}_C^-(\mathcal{A}) = \{\mathtt{M}_2, \mathtt{M}_4\}$, a good definition for $C = \mathtt{SendBack}$ may be:

$$\mathtt{SendBack} \equiv \mathtt{Machine} \sqcap \exists \mathtt{hasPart}.(\mathtt{Worn} \sqcap \neg \mathtt{Replaceable})$$

A more general setting may be conceived to manage the case of refinement problems, in which a definition for the target concept is already available but it may be defective w.r.t. to some positive or negative examples [7, 6]. Note that this task differs also from settings where the aim is building a classifier, rather than DL concept definitions, through parametric and non-parametric statistical methods [12, 13, 14]. These related settings will not be further discussed.

4 Terminological Decision Trees and Their Induction

First-order logical decision trees (FOLDTs) are defined [9] as binary decision trees in which

1. the nodes contain tests in the form of conjunctions of literals;
2. left and right branches stand, resp., for the truth-value (resp. true and false) determined by the test evaluation;
3. different nodes may share variables, yet a variable that is introduced in a certain node must not occur in the right branch of that node.

Terminological decision trees (TDTs) extend the original definition, allowing DL concept descriptions as (variable-free) node tests. Fig. 1 shows a TDT denoting also the definition of the `SendBack` concept proposed in Ex. 3.1.

4.1 Classification

The TDTs can be used for classifying individuals. Fig. 2 shows the related classification procedure. It uses other functions: LEAF() to determine whether a node is a leaf of the argument tree, ROOT() which returns the root node of the input tree, and INODE() which retrieves the test concept and the left and right subtrees branching from a given internal node. Given an individual a, starting from the root node, the algorithm checks the class-membership w.r.t. the test concept D_i in the current node, i.e. $\mathcal{K} \models D_i(a)$, sorting a to the left branch if the test is successful while the right branch is chosen if $\mathcal{K} \models \neg D_i(a)$. Eventually the classification is found as a leaf-node concept.

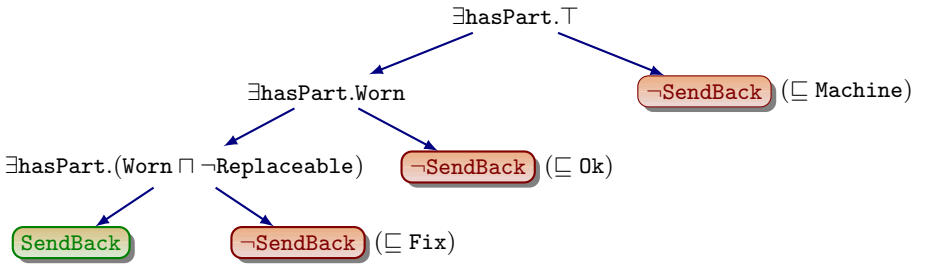

Fig. 1. A TDT whose leftmost path corresponds to the DL concept definition `SendBack` $\equiv \exists$`hasPart.(Worn` $\sqcap \neg$`Replaceable)`. Other definitions can be associated to the paths to leaves labeled with $\neg$`SendBack` that are related to other (disjoint) concepts.

function CLASSIFY(a: individual, T: TDT, $\mathcal{K}$: KB): concept;
begin

1. $N \leftarrow$ ROOT(T);
2. **while** $\neg$LEAF(N, T) **do**
   (a) $(D, T_{\text{left}}, T_{\text{right}}) \leftarrow$ INODE(N);
   (b) **if** $\mathcal{K} \models D(a)$ **then** $N \leftarrow$ ROOT(T_{left})
   (c) **elseif** $\mathcal{K} \models \neg D(a)$ **then** $N \leftarrow$ ROOT(T_{right})
   (d) **else return** $\top$
3. $(D, \cdot, \cdot) \leftarrow$ INODE(N);
4. **return** D;

end

Fig. 2. Classification with TDTs

Note that the open-world semantics may cause `unknown` answers (failure of both left and right branch tests) that can be avoided by considering a weaker (default) right-branch test: $\mathcal{K} \not\models D_i(a)$. This differs from the FOLDTs where the test actually consists of several conjunctions that occur in the path from the root to the current node.

4.2 From Terminological Decision Trees to Concept Descriptions

Note that each node in a path may be used to build a concept description through specializations. This can be given 1) by adding a conjunctive concept description, 2) by refining a sub-description in the scope of an existential, universal or number restriction or 3) by narrowing a number restriction (which may be allowed by the underlying language, e.g. $\mathcal{ALN}$ or $\mathcal{ALCQ}$). No special care[8] is to be devoted to negated atoms and their variables. The underlying tableau reasoning algorithm for reasoning with $\mathcal{ALC}$ is indeed sound and complete (see [1], Ch. 3).

[8] We are considering expressive (and decidable) DL languages like $\mathcal{ALCQ}$, that are endowed with full negation, hence the situation is perfectly symmetric.

function DERIVEDEFINITION(C: concept name, T: TDT): concept description;
begin

1. $S \leftarrow$ ASSOCIATE$(C, T, \top)$;
2. **return** $\bigsqcup_{D \in S} D$;

end

function ASSOCIATE(C: concept name; T: TDT; D_c: current concept description): set of descriptions;
begin

1. $N \leftarrow$ ROOT(T);
2. $(D_n, T_{\text{left}}, T_{\text{right}}) \leftarrow$ INODE(N);
3. **if** LEAF(N, T)
   then
   (a) **if** $D_n = C$ **then**
   return $\{D_c\}$;
   else
   return $\emptyset$;
   else
   (a) $S_{\text{left}} \leftarrow$ ASSOCIATE$(C, T_{\text{left}}, D_c \sqcap D_n)$;
   (b) $S_{\text{right}} \leftarrow$ ASSOCIATE$(C, T_{\text{right}}, D_c \sqcap \neg D_n)$;
   (c) **return** $S_{\text{left}} \cup S_{\text{right}}$;

end

Fig. 3. Mapping a TDT onto a DL concept description

For each target concept name C it is possible to derive a *single* concept definition from a TDT. The algorithm (see Fig. 3) follows all the paths leading to success nodes i.e. leaves labeled with C (the heads of the clauses in the original setting) collecting the intermediate test concepts (formerly, the body literals). In this way, each path yields a different conjunctive concept description. that represents a different version of the target concept in conjunctive form $D_i = D_1^i \sqcap \cdots \sqcap D_l^i$. The final single description for the target concept is obtained as the disjunctive description built with concepts from this finite set $S = \{D_i\}_{i=1}^M$. Hence, the final definition is $C \equiv \bigsqcup_{i=1}^M D_i$. As an example, looking back at the TDT depicted in Figure 1, a concept definition that may be extracted is

$$\texttt{Ok} \equiv \exists \texttt{hasPart}.\top \sqcap \neg \exists \texttt{hasPart}.\texttt{Worn} \equiv \exists \texttt{hasPart}.\top \sqcap \forall \texttt{hasPart}.\neg \texttt{Worn}$$

i.e. something that has exclusively parts which are not worn.

Like in the original logic tree induction setting, also internal nodes may be utilized to induce new intermediate concepts.

4.3 Induction of TDTs

The subsumption relationship $\sqsubseteq$ induces a partial order on the space of DL concept descriptions. Then, the learning task can be cast as a search for a solution

of the problem in the partially ordered space. In such a setting, suitable operators to traverse the search space are required [4, 6].

While existing DL concept induction algorithms generally adopt a separate-and-conquer covering strategy, the TDT-learning algorithm adopts a divide-and-conquer strategy [10]. It also tries to cope with the limitations of the other learning systems, namely approximation and language-dependence. Indeed, since the early works [2], instances are required to be transposed to the concept level before the learning can start. This is accomplished by resorting to the computation, for each training individual, of the related MSC the individual belongs to [1], which need not exist, especially for expressive DLs, and thus has to be approximated. Even in an approximated version, the MSCs turn out to be extremely specific descriptions which affects both the efficiency of learning and the effectiveness of the learned descriptions as this specificity easily leads to overfitting the data [4].

The algorithms implemented by DL-Learner [6] partly mitigate these disadvantages being based on stochastic search using refinement operators and a heuristic computed on the grounds of the covered individuals (and a syntactic notion of concept length). Generate-and-test strategies may fall short when considering growing search spaces determined by more expressive languages. This drawback is hardly avoidable and it has been tackled by allowing more interaction with the knowledge engineer which can be presented with partial solutions and then decide to stop further refinements.

Our TDT-induction algorithm adapts the classic schema implemented by C4.5 [8] and Tilde [9]. A sketch of the main routine is reported in Fig. 4. It reflects the standard tree induction algorithms with the addition of the treatment of unlabeled training individuals. The three initial conditional statements take care of the base cases of the recursion, namely:

1. no individuals got sorted to the current subtree root then the resulting leaf is decided on the grounds of the prior probabilities of positive and negative instances (resp. Pr_+ and Pr_-);
2. no negative individual yet a sufficient rate (w.r.t. the threshold θ) of positive ones got sorted to the current node, then the leaf is labeled accordingly;
3. dual case w.r.t. to the previous one.

The second half of the algorithm (randomly) generates a set *Specs* of (satisfiable) candidate descriptions (calling generateNewConcepts), that can specialize the current description D when added as a conjunction. Then, the best one (D_{best}) is selected in terms of an improvement of the purity of the subsets of individuals resulting from a split based on the test description. The (im)purity measure is based on the entropic *information gain* [8] or on the *Gini index* which was finally preferred. In the DL setting the problem is made more complex by the presence of instances which cannot be labelled as positive or negative (see [7]) whose contributions are considered as proportional to the prior distribution of positive and negative examples.

Once the best description D_{best} has been selected (calling selectBestConcept), is is installed as the current subtree root and the sets of individuals sorted

function INDUCETDTREE(C: concept name; D: current description;
Ps, Ns, Us: set of (positive, negative, unlabeled) training individuals): TDT;
const θ; {purity threshold}
begin
Initialize new TDT T;
if $|Ps| = 0$ **and** $|Ns| = 0$
then {*pure* leaf node}
if $Pr_+ \geq Pr_-$
then T.root $\leftarrow C$
else T.root $\leftarrow \neg C$;
return T;
if $|Ns| = 0$ **and** $|Ps|/(|Ps| + |Ns| + |Us|) > \theta$ **then**
begin T.root $\leftarrow C$; **return** T; **end**
if $|Ps| = 0$ **and** $|Ns|/(|Ps| + |Ns| + |Us|) > \theta$ **then**
begin T.root $\leftarrow \neg C$; **return** T; **end**
$Specs \leftarrow$ GENERATENEWCONCEPTS(D, Ps, Ns);
$D_{best} \leftarrow$ SELECTBESTCONCEPT($Specs$, Ps, Ns, Us);
$((P^l, N^l, U^l), (P^r, N^r, U^r)) \leftarrow$ SPLIT(D_{best}, Ps, Ns, Us);
T.root $\leftarrow D_{best}$;
T.left $\leftarrow$ INDUCETDTREE($C, D \sqcap D_{best}, P^l, N^l, U^l$);
T.right $\leftarrow$ INDUCETDTREE($C, D \sqcap \neg D_{best}, P^r, N^r, U^r$);
return T;
end

Fig. 4. The main routine for inducing terminological decision trees

to this node are subdivided according to their classification w.r.t. such a concept. Note that unlabeled individuals must be sorted to both subtrees. Finally the recursive calls for the construction of the subtrees are made, passing the proper sets of individuals and the concept descriptions $D \sqcap D_{best}$ and $D \sqcap \neg D_{best}$ related to either path.

5 Experimental Evaluation

5.1 Experimental Setting

To test the new algorithm on real ontologies, the resulting system TERMITIS was applied to a number of individual classification problems solved by inducing TDTs w.r.t. random query concepts. To this purpose, a number of ontologies represented in OWL concerning different domains have been selected[9], namely: FINITESTATEMACHINES (FSM) concerning finite state machines, NEWTESTAMENTNAMES (NTN) accounting for characters and places mentioned in that

[9] The ontologies can be found in standard repositories: the Protégé library (`http://protege.stanford.edu/plugins/owl/owl-library`) and TONES (`http://owl.cs.manchester.ac.uk/repository`).

Table 1. Facts concerning the ontologies employed in the experiments

ontology	DL language	#concepts	#object properties	#datatype properties	#individuals
FSM	$\mathcal{SOF}(\mathcal{D})$	20	10	7	37
MDM0.73	$\mathcal{ALCHOF}(\mathcal{D})$	196	22	3	112
WINES	$\mathcal{ALCOF}(\mathcal{D})$	75	12	1	161
BIOPAX	$\mathcal{ALCIF}(\mathcal{D})$	74	70	40	323
HDISEASE	$\mathcal{ALCIF}(\mathcal{D})$	1498	10	15	639
NTN	$\mathcal{SHIF}(\mathcal{D})$	47	27	8	676
FINANCIAL	$\mathcal{ALCIF}$	60	16	0	1000

book, the BioPax glycolysis ontology (BioPax) describing the glycolysis pathway from the EcoCyc database, the WINE ontology from a project describing Wines and Food, the FINANCIAL ontology, built for eBanking applications and two medical ontologies MDM0.73 and HDISEASE. Tab. 1 summarizes important details concerning these ontologies, in terms of the numbers of concepts, object and datatype properties and individuals. The sizes of the ontologies are to be measured in terms of (thousands of) RDF triples.

Artificial learning problems were created generating 50 random queries per ontology by composition of 2 through 8 concepts built by means of the language constructors: complement ($\neg$), intersection ($\sqcap$), union ($\sqcup$), universal ($\forall$) or existential ($\exists$) restrictions. The general experiment design adopted a *.632 bootstrap* strategy. A standard reasoner[10] was employed to decide on the real class-membership (and non-membership) w.r.t. the query concepts.

The performance was evaluated comparing the classification of the test individuals w.r.t. the query concepts performed using both the induced trees and the deductive instance-checking provided by the reasoner. The prior distribution of positive and negative instances were computed for each ontology. The default setting of the threshold ($\theta = .05$) was considered.

5.2 Outcomes

Due to the open-world semantics, it may happen that the membership of an individual w.r.t. a query concept cannot be determined by a reasoner (see Sect. 2). Then a three-way classification problems were considered and the induced model was evaluated using the following indices also for allowing a comparison to the outcomes of experiments with another DL concept learning system like DL-FOIL [7]. Essentially they measure the correspondence between the deductive and inductive classification for the test instances w.r.t. the query concepts provided, resp., using a DL reasoner and the TDT algorithm (see Fig. 2):

- *match rate*, i.e. number of cases of individuals that got the same classification with both modes;

[10] PELLET ver. 2 (available at `http://clarkparsia.com/pellet`).

Table 2. Results: average values ± standard deviations

ontology	match rate	commission rate	omission rate	induction rate
FSM	96.68±01.98	00.99±01.35	00.02±00.18	02.31±00.51
MDM0.73	93.96±05.44	00.39±00.61	03.50±04.16	02.15±01.47
WINES	74.36±25.63	00.67±04.63	12.46±14.28	12.13±23,49
BIOPAX	96.51±06.03	01.30±05.72	02.19±00.51	00.00±00,00
HDISEASE	78.60±39.79	00.02±00.10	01.54±06.01	19.82±39.17
NTN	91.65±15.89	00.01±00.09	00.36±01.58	07.98±14.60
FINANCIAL	96.21±10.48	02.14±10.07	00.16±00.55	01.49±00.16

- *omission error rate*, amount of individuals for which the membership w.r.t. the given query could not be determined using the TDT, while they can be proven to belong to the query concept or to its complement;
- *commission error rate*, amount of individuals found to belong to the query concept according to the induced TDT, while they can be proven to belong to its complement and vice-versa;
- *induction rate*, amount of individuals found to belong to the query concept or its complement according to the TDT, while either case is not logically derivable from the knowledge base.

Tab. 2 reports the outcomes in terms of these indices. Preliminarily, we found that the search procedure was accurate enough: it made few critical mistakes, especially when the considered concepts are known to have many examples (and counterexamples) in the ontology. However, it is important to note that, in each experiment, the commission error was limited but not absent, as in the experiments with other classification methods. The cases of queries for which this measure was high are due to the limited amount of examples available (concepts with narrow extensions). Even few mistakes provoked high error rates. This is also due to the absence of axioms stating explicitly the disjointness between some concepts. Also the omission error rates are quite low. They are comparable with the amount of inductive conclusions that could be drawn with the induced definitions. Again these figures may vary as a consequence of the presence / absence of knowledge about the disjunction of (sibling) concepts in the subsumption hierarchies. In an ontology population perspective, the cases of induction are interesting because they suggest new assertions which cannot be logically derived by using a deductive reasoner yet they might be used to complete a knowledge base, e.g. after being validated by an ontology engineer and/or a domain expert. Better results were obtained on the same task with different inductive methods. Yet, like with DL-FOIL we have the added value of having an intensional definition of the target concepts.

The elapsed time was very limited: about 0.5 hour for a whole experiment on a medium sized ontology (in terms of number of individuals) including the time consumed by the reasoner for the deductive instance checks.

BioPax
induced concept:
```
(Or (And physicalEntity protein) dataSource)
```
original concept:
```
(Or (And (And dataSource externalReferenceUtilityClass)
         (ForAll ORGANISM (ForAll CONTROLLED phys icalInteraction)))
     protein)
```

NTN
induced conceot:
```
(Or EvilSupernaturalBeing (Not God))
```
original concept:
```
(Not God)
```

Financial
induced concept:
```
(Or (Not Finished) NotPaidFinishedLoan Weekly)
```
original concept:
```
(Or LoanPayment (Not NoProblemsFinishedLoan))
```

Fig. 5. Examples of induced description for given target concepts employed for the generation of training examples

A comparison with previous works is difficult because some of them consider only binary problems so that standard accuracy measures can be applied. However, as the unknown membership is inherent in the semantics of the representation, it appears to be fairer to consider ternary problems distinguishing counter-examples from those of uncertain membership. As we used the same measures employed for the evaluation of DL-Foil [7], some comparison can be made with that system (although a 10 fold cross-validation design was adopted there). For all the three ontologies employed in the former experiment (BioPax, NTN and Financial) the performance was improved (from 75% to over 90% match rates while the commission errors, formerly up to 16-19% are reduced to a few percentage points). Also in terms of variance, there is a significant reduction of the standard deviation, a sign that the new method is also more stable.

5.3 Qualitative Evaluation

For some of the ontologies, we report in Figure 5 some examples of the concept descriptions (in a Lisp-like syntax) that were learned during the experiments and compare them to the original query concept that generated the examples and counterexamples.

The same concepts have been processed by DL-Foil with the same results, while former systems (such as YinYang [4]) tended to output accurate yet unnecessarily long descriptions. Of course for a correct qualitative interpretation of the value of these concepts some familiarity is assumed with the domain of

the ontologies. However we notice that the induced concepts generally tend to be slightly more complex than the original descriptions. This is not so when learning from MSCs [2, 4]. Moreover, they are correct w.r.t. the individuals occurring in the ontology: the extensions of the original and induced concepts, measured in terms of the known instances (retrieval set), generally overlap.

6 Conclusions and Outlook

Introducing terminological decision trees, we investigated new methods (based on logical decision trees) for learning concepts in expressive DLs representations that support the standard Web ontology languages. In the TermiTIS system, a top-down tree induction algorithm was implemented that is an adaptation of standard tree induction methods to the issues related to the different representation. Indeed, as with the predecessor DL-Foil, it essentially makes use of a different gain function which takes into account the open-world semantics. This requires a different setting and a special treatment of the unlabeled individuals which is similar to the semi-supervised learning problem [11] in a FOL context.

The presented experimental evaluation, applying the TermiTIS system to the task of individual classification real ontologies (some had been already used for experimenting DL-Foil) using the same performance indices, measuring the alignment of the classifications decided by the induced TDTs with those derived deductively by a DL reasoner. This allowed measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their negations), which constituted a real added value brought by the inductive method. The experiments made on various ontologies showed that the method is quite effective, and its performance depends on the number (and distribution) of the available training individuals. Besides, the procedure appears also robust to noise since commission errors were limited.

Actually this sort of *abductive* assertions produced by the inductive method should be evaluated by domain experts (e.g. those who supported the construction of the ontology). Namely, validated assertions may be employed in the task of ontology population. This also allows for a more focused diagnosis of the ontologies as it may elicit specific parts that would require some amendment.

We plan to extend this work in various directions. First of all the underlying DL language is being extended. The method can already manage KBs represented in more expressive languages than $\mathcal{ALCQ}$ but use the concepts therein as atoms and building new ones exclusively thorough $\mathcal{ALCQ}$ concept constructors. More impurity indices have to be explored especially to better take into account the uncertainty related to the unlabeled individuals.

Finally, the presented method may be the basis for alternative hierarchical clustering algorithms where clusters would be formed grouping individuals on the grounds of the invented subconcept instead of their similarity, as this may be hardly defined with such complex representations.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)

[2] Cohen, W., Hirsh, H.: Learning the CLASSIC description logic. In: Torasso, P., et al. (eds.) Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning, pp. 121–133. Morgan Kaufmann, San Francisco (1994)

[3] Badea, L., Nienhuys-Cheng, S.H.: A refinement operator for description logics. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000)

[4] Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26, 139–159 (2007)

[5] Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Concept formation in expressive description logics. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 99–113. Springer, Heidelberg (2004)

[6] Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning 78, 203–250 (2010)

[7] 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)

[8] Quinlan, R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)

[9] Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artificial Intelligence 101, 285–297 (1998)

[10] Boström, H.: Covering vs. divide-and-conquer for top-down induction of logic programs. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995, pp. 1194–1200. Morgan Kaufmann, San Francisco (1995)

[11] Goldman, S.A., Kwek, S., Scott, S.D.: Learning from examples with unspecified attribute values. Information and Computation 180, 82–100 (2003)

[12] d'Amato, C., Fanizzi, N., Esposito, F.: Query answering and ontology population: An inductive approach. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 288–302. Springer, Heidelberg (2008)

[13] Fanizzi, N., d'Amato, C., Esposito, F.: Statistical learning for inductive query answering on OWL ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 195–212. Springer, Heidelberg (2008)

[14] Luna, J.O., Cozman, F.G.: An algorithm for learning with probabilistic description logics. In: Bobillo, F., et al. (eds.) Proceedings of the 5th International Workshop on Uncertainty Reasoning for the Semantic Web, URSW2009. CEUR Workshop Proceedings, vol. 527, pp. 63–74. CEUR-WS.org (2009)