Appl Intell (2007) 26:139–159
DOI 10.1007/s10489-006-0011-5

An algorithm based on counterfactuals for concept learning in the Semantic Web

Luigi Iannone · Ignazio Palmisano · Nicola Fanizzi

Published online: 18 January 2007

Abstract In the line of realizing the Semantic-Web by means of mechanized practices, we tackle the problem of building ontologies, assisting the knowledge engineers' job by means of Machine Learning techniques. In particular, we investigate on solutions for the induction of concept descriptions in a semi-automatic fashion. In particular we present an algorithm that is able to infer definitions in the $\mathcal{ALC}$ Description Logic (a sub-language of OWL-DL) from instances made available by domain experts. The effectiveness of the method with respect to past algorithms is also empirically evaluated with an experimentation in the *document image understanding* domain.

Keywords Ontology learning · Refinement operators · Inductive reasoning · Machine learning · Knowledge management · Ontology evolution

1 Introduction

1.1 The Scenario

Since Tim Berners-Lee coined the name Semantic Web (SW) for his personal vision of the brand new Web [6], a lot of effort has been spent by research community—especially Knowledge Representation & Reasoning (KRR) groups—in finding out the most suitable formalism for representing knowledge in the Semantic Web. SW, in fact, stands as a stack of specifications for KR languages in order to make information (or, better, knowledge) directly processable by machines. This stack relies on very well assessed pre-existing Web technologies such as XML[1] or URI[2] and builds up on these standards a framework for representing metadata in order to enrich existing resources on the Web (RDF[3]) with that. In order to be interoperable, the metadata annotations should be expressed in terms of shared vocabularies, where they are defined with their properties and with the relationships among them. The evolution of standard specification for expressing such *vocabularies* started from RDFSchema [27] and moved to OWL (Web Ontology Language) [21], making it clear that the SW choice in representing metadata concepts and relationships, and reasoning with them, was Description Logics (DLs) [2]. The term *ontology* was borrowed from philosophy, but with a different meaning, which can be summarized as "a specification of a conceptualization" [19].

In the evolution track of SW, already consolidated, we currently have specifications to write down portable documents (XML), to enrich them (or other kinds of documents, e.g. HTML pages) with metadata (RDF), and to build metadata ontologies, that are collections of taxonomic and non taxonomic relationships among metadata classes. Since these ontologies are based on Description Logics, they have formal semantics and, hence, it is possible to implement inferences on ontology based representations.

L. Iannone (✉) · I. Palmisano · N. Fanizzi
Dipartimento di Informatica – Università degli Studi di Bari, Via Orabona 4, 70125 Bari, Italy
e-mail: iannone@di.uniba.it

I. Palmisano
e-mail: palmisano@di.uniba.it

N. Fanizzi
e-mail: fanizzi@di.uniba.it

[1] eXtensible mark-up language http://www.w3.org/XML

[2] Uniform Resource Identifiers http://www.w3.org/Addressing/

[3] Resource Description Framework - http://www.w3.org/RDF

1.2 Motivations

In this big picture, we proceed now illustrating the rising issues that we intend to tackle. Semantic Web ensures semantic interoperability by means of ontologies that specify the intended meaning of metadata in terms of their relationships with the entities making up their domain. The problem is that, nowadays, the Web has not yet been provided with a considerable number of ontologies. There are few of them available, on very few subjects. Moreover, building up an ontology from scratch can be a very burdensome and difficult task [24], and two knowledge engineers would produce different ontologies for the same domain each reflecting its creator's personal view. Though these differences could appear trivial for humans, they can depend on various factors (level of granularity, different points of view), and cannot be easily harmonized by machines. Therefore, we need an approach for building ontologies that is:

– (at least) semi-automated.
– predictable and controllable, in the sense that, fixed some input parameters, it does not produce different results at each run on the same domain.

We argue that a Machine Learning approach is very appropriate in this setting, providing both the necessary flexibility and the formal support for automatically acquiring ontologies. In particular, we study the problem of building up a concept definition starting from positive and negative examples of the target concept itself. This is the line of past [9] and present [3, 14] research aiming at finding methods for the bottom-up construction of knowledge bases. It could be argued that the standard practice of ontology engineering concepts and properties are devised before populating the knowledge base with instances. Yet, it is our opinion that an effective approach would consist in asking the expert for assertions on individuals, given in terms of primitive concepts, and then run a system to induce more complex concepts to be added to the ontology on the input given by the expert.

Additionally, another practical situation in which the approach we illustrate in this work can reveal itself useful is during ontology repair and evolution when refined or new concept definitions are to be constructed. Knowledge engineers can follow two approaches:

– Writing the new intensional definitions of the concepts in the desired ontology language (e.g. OWL).
– Use a supervised machine learning algorithm for inducing the new concept definition by providing examples and counterexamples[4] of the target concept.

[4] Elsewhere in literature and in this paper called also, respectively, positive and negative examples.

Though the first solution could appear simpler, it may hide some undesirable drawbacks. For instance, none can guarantee that the new definition is consistent with the world state already described in the knowledge base. Moreover, in writing the definition engineers could miss features that do not emerge until they look at the data. Although logic methods for dealing with inconsistent (or incomplete) knowledge bases exist, these add their overhead to reasoning that cannot be always justified in real-life scenarios. Indeed, incorrect knowledge bases are also the target of diagnosis techniques which could suggest those parts in the ontologies that need a refinement.

A practical case could be the extension of an interesting work by Astrova [1] in which the author proposes a methodology for building an ontology from a relational database. Let us suppose that after the process one realizes that the resulting ontology lacks some classes (concepts) that can be built starting from the basic ontology formerly obtained. Instead of writing the missing definition from scratch, the knowledge engineer may properly select some positive and negative examples (that in this case are no more that tuples of some view in the database) and run a concept learning algorithm like the one we will propose.

This being the scenario and motivations, we proceed now illustrating the rising issues that we want to tackle, recalling related approaches to such problems (Section 2). The search space is reviewed in Section 3 and the refinement operators that traverse this space and their properties are presented in Section 4. Then we illustrate our solution from a theoretical point of view (Section 5), we provide a practical example and an empirical evaluation of our approach (Section 6). Finally, (Section 7) some conclusions are drawn and further enhancements to this work are presented.

2 Related work

Some approaches to the problem of learning concept definitions in DL formalisms are already present in literature; one of the distinction between them is whether the problem is solved translating to another formalism in which concept learning has already been investigated, or attacking it in the original formalism.

An example of the first approach can be found in [23], where Kietz refers to an hybrid language called CARIN-$\mathcal{ALN}$ [28]. Such language combines a complete structural subsumption endowed DL with Horn logic, where terms are individuals in the domain. Reasoning for such formalism is shown to be equivalent to subsumption in DL and the learnability boundaries for polynomial learning (in the sense of Valiant [31]) are explored. The problem translation turns out to be similar to an Inductive Learning Programming (ILP) problem; Kietz shows that subsumption can be accomplished by means of θ-subsumption and lcs using a variant of Plotkin's *Least General Generalizer* - lgg; moreover, the author proves that CARIN-$\mathcal{ALN}$ is Probably Approximately Correct learnable (PAC -learnable). There are two issues on this kind of approach: there are incompatibilities among DLs and Horn Logic, that in the case of $\mathcal{ALN}$ are in the impossibility of translating background knowledge axioms like:

$$\forall R.D \sqsubseteq C$$

and the fact that the Closed World Assumption is not used in DL (reasoning with Open World Assumption); as a consequence, none of the at-most learned restrictions would cover the positive instances if we used DL reasoners.

An example of the other approach, i.e. solving the learning problem in the original formalism, can be found in [9], devised a pure DL-based approach for concept learning. Cohen and Hirsh work is based on the CLASSIC DL, which provides a special constructor (namely SAME-AS) that can simulate the variable *linkedness* in Horn-clauses with the restriction on functional roles[5] in order to keep this DL decidable. Such learnability evaluation gave polynomial learnability for its k-CLASSIC sublanguage, obtained by fixing an arbitrary integer k representing the maximum number of attributes in a SAME-AS chain [8].

The learning problem differs from the previous one because positive and negative examples are DL concepts that are very specific representative at concept level of the corresponding instances. This is due to the fact that DL has two different languages, one for concepts and another one for instances, and represents the application of the *Single representation trick* by Dietterich et al. in [12]. On the other side, practical handling of inconsistent/incomplete ontologies is mainly dealt with assuming there are wrong axioms or wrong information in the assertions. On the base of this assumption, the work by [18] presents an algorithm to isolate the minimal set of axioms introducing inconsistency in a knowledge base, thus isolating the portion of the ontology that the knowledge engineer has to review; however, this approach is not useful in cases in which the ontology is consistent but misses useful definitions, which is a common situation when handling ontology evolution as defined in [20], where the authors introduce a categorization of ontology evolution processes and related tasks, e.g. ensuring consistency whenever a change in ontology happens or obtaining correct answers when querying an inconsistent ontology. However, these approaches are not aimed at building new concept definitions, but at debugging and maintaining existing ontologies.

[5] DL roles admitting only one filler per individuals likewise functions.

3 The search space

In this section we illustrate the space of concept definitions that our methods have to explore in the quest for correct ones. In particular we show our results for a specific DL named $\mathcal{ALC}$. DLs differ from each other for the constructs they allow. In order to make this paper as self-contained as possible we report syntax and semantics for $\mathcal{ALC}$; for a more thorough description please refer to [29].

In every DL, primitive *concepts* $N_C = \{C, D, \ldots\}$ are interpreted as subsets of a certain domain of objects (resources) and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain (properties). More complex concept descriptions are built using atomic concepts and primitive roles by means of the constructors in Table 1.

Their meaning 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.

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

Definition 3.1 (subsumption). Given two concept descriptions C and D in $\mathcal{T}$, C *subsumes* D, denoted by $C \sqsupseteq D$, iff for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$. Hence, $C \equiv D$ amounts to $C \sqsupseteq D$ and $D \sqsupseteq C$. Whereas $C \sqsupset D$ means that for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \supset D^{\mathcal{I}}$.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a T-box $\mathcal{T}$ and an A-box $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions $C \equiv D$, meaning $C^{\mathcal{I}} = D^{\mathcal{I}}$, where C is the concept name and D is a description given in terms of the language constructors. Actually, there exist general T-Boxes that allow also for axioms like $C \sqsubseteq D$ or $C \sqsupseteq D$ and for cycles in definition, but in this paper we restrict to what in literature are called *acyclic* T-Boxes in which there are only concept definitions. Such definitions are in the form *ConceptName* $\equiv D$ (one can easily show that they are equivalent to acyclic T-Boxes with complex concepts on both sides of equivalence sign). $\mathcal{A}$ contains extensional assertions on concepts, roles and individuals, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (the interpretation of the individual called a belongs to the interpretation of the concept C) and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ (the couple a, b belongs to the interpretation of the relationship R).

Example 3.1. A possible concept definition in the proposed language is:
$\mathsf{Father} \equiv \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists \mathsf{hasChild.Human}$
which translates the sentence: *"a father is a male human that has some humans as his children"*.

Table 1 The constructors for $\mathcal{ALC}$ descriptions and their interpretation

NAME	SYNTAX	SEMANTICS
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x, y) \in R^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$

A-box assertions look like:
Father(*Tom*), *Father*(*Bill*), *hasChild.Human*(*Bill*, *Joe*) and so on.

Now, if we define two new concepts:
$\mathsf{FatherWithoutSons} \equiv \mathsf{Human} \sqcap \mathsf{Male} \sqcap$
$\exists \mathsf{hasChild.Human} \sqcap \forall \mathsf{hasChild.}(\neg \mathsf{Male})$
$\mathsf{Parent} \equiv \mathsf{Human} \sqcap (\mathsf{Male} \sqcup \mathsf{Female}) \sqcap \exists \mathsf{hasChild.Human}$
then it is easy to see that
$\mathsf{Father} \sqsupseteq \mathsf{FatherWithoutSons}$
and also that
$\mathsf{Parent} \sqsupseteq \mathsf{Father}$,
yet
$\mathsf{Father} \not\sqsupseteq \mathsf{Parent}$ and $\mathsf{FatherWithoutSons} \not\sqsupseteq \mathsf{Father}$

Preliminarily, some notation is needed to name the different parts of an $\mathcal{ALC}$ description: $\mathsf{prim}(C)$ is the set of all the concepts at the top-level conjunction of C; if there exists a universal restriction $\forall R.D$ on the top-level of C then $\mathsf{val}_R(C) = \{D\}$ (a singleton description because many such restrictions can be collapsed into a single one with a conjunctive filler concept) otherwise $\mathsf{val}_R(C) = \{\top\}$. Finally, $\mathsf{ex}_R(C)$ is the set of the concept descriptions E appearing in existential restrictions $\exists R.E$ at the top-level conjunction of C.

Definition 3.2 ($\mathcal{ALC}$ normal form). A concept description D is in $\mathcal{ALC}$ *normal form* iff $D \equiv \bot$ or $D \equiv \top$ or if $D = D_1 \sqcup \cdots \sqcup D_n$ with

$$D_i = \bigsqcap_{A \in \mathsf{prim}(D_i)} A \sqcap \bigsqcap_{R \in N_R} \left[\bigsqcap_{V \in \mathsf{val}_R(D_i)} \forall R.V \right] \sqcap \left[\bigsqcap_{E \in \mathsf{ex}_R(D_i)} \exists R.E \right]$$

where, for all $i = 1, \ldots, n$, $D_i \not\equiv \bot$ and for any R, every sub-description in $\mathsf{ex}_R(D_i)$ and $\mathsf{val}_R(D_i)$ is in normal form.

4 Induction and refinement in $\mathcal{ALC}$

Notice that subsumption imposes a partial order relationship on any set of DL concepts. In the following, in fact, we will consider a set of concepts definition ordered by subsumption as a search space $(\mathcal{S}, \succeq)$ in which the algorithm has to find out a consistent definition for the target concept. Our problem of induction in its simplest form can be now formally defined as a supervised learning task:

Definition 4.1 (learning problem). In a search space $(\mathcal{S}, \succeq)$
Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of positive and negative assertions $\mathcal{A}_C = \mathcal{A}_C^+ \cup \mathcal{A}_C^-$ regarding the membership (or non-membership) of some individuals to a concept C such that: $\mathcal{A} \not\models_{\mathcal{T}} \mathcal{A}_C$
Find a new T-box $\mathcal{T}' = (\mathcal{T} \setminus \{C \equiv D\}) \cup \{C \equiv D'\}$ such that: $\mathcal{A} \models_{\mathcal{T}'} \mathcal{A}_C$

Thus, if a concept C is not defined in the terminology $\mathcal{T}$ we have a case of an *induction problem* requiring to find definitions $C \equiv D$ entailing the (new) assertions in $\mathcal{A}_C$. Conversely, when an existing definition in $\mathcal{T}$ proves to be incorrect i.e. it is not capable of entailing the positive assertions in $\mathcal{A}_C$ (*incomplete* definition) or it entails negative ones (*inconsistent* definition), this yields a *refinement problem* where a new correct definition $C \equiv D'$ is to be found on the ground of the previous one and the new examples.

Both problems can be cast as search problem on the space of all possible concept definitions in the given $\mathcal{ALC}$ DL. In order to traverse this space one needs operators that allow for moving across concepts in two directions (as we said that this kind of spaces are ordered by subsumption) [26]. In fact, given a concept definition in such a space one can:

– Obtain a more general one (upward refinement operator).
– Obtain a more specific one (downward refinement operator).

Depending on the target DL one can imagine many refinement operators. Typically they are operators that manipulate the syntactical structure of the concept that has been previously put in a particular *normal form*. This kind of syntactical way of re-writing concepts, usually, presents some useful features that result in simplified refinement operators, and exists for any concept description in the target DL. Some examples of refinement operators can be found in [4] though for different DL, namely $\mathcal{ALER}$.

Let us define refinement operators, respectively ρ and δ, for specialization and generalization.

Definition 4.2 (downward refinement operator—ρ). Let $(\mathcal{S}, \preceq)$ be a search space of concept descriptions C ordered by subsumption relation $\sqsubseteq$ (see Def. 3.1) we define a *downward refinement operator* as the function $\rho : \mathcal{S} \to 2^{\mathcal{S}}$ such that $\rho(C) \subseteq \{E \in \mathcal{S} \mid E \sqsubseteq C\}$.

Definition 4.3 (upward refinement operator—δ). Let $(\mathcal{S}, \preceq)$ be a search space of concept descriptions C ordered by subsumption relation $\sqsubseteq$ (see Def. 3.1) we define an *upward refinement operator* as the function $\delta : \mathcal{S} \to 2^{\mathcal{S}}$ such that $\delta(C) \subseteq \{E \in \mathcal{S} \mid C \sqsubseteq E\}$.

Notice that it does not hold in general either that $E \sqsubseteq C \to E \in \rho(C)$ or $C \sqsubseteq E \to E \in \delta(C)$.

Definition 4.4 (refinement chain). A downward (upward) refinement chain from C to D is a sequence $C_0, C_1, \ldots, C_n \equiv D$ such that $C_n \in \rho(C_{n-1})$ (respectively $C_n \in \delta(C_{n-1})$, $1 \leq i \leq n$).

Definition 4.5 (downward/upward cover). D is a downward (upward) cover of C iff $D \sqsubset C$ ($C \sqsubset D$) and it does not exist E such that $D \sqsubset E \sqsubset C$ ($C \sqsubset E \sqsubset D$).

Definition 4.6 (refinement properties). Let $\mathcal{S}$ be a search space ordered by subsumption. A refinement operator ρ (δ) is

- locally finite if $\forall C \in \mathcal{S} \rho(C)$ ($\delta(C)$) has finite cardinality
- weakly complete if $\mathcal{S} \subseteq \rho^*(\top)$ ($\mathcal{S} \subseteq \delta^*(\bot)$)
- complete if $\forall C, D$ such that $D \sqsubset C$ then $\exists E \in \rho^*(C), E \equiv D$ ($\exists E \in \delta^*(D), E \equiv C$)
- minimal if $\forall D \in \rho(C) (D \in \delta(C)) : D$ is a downward (upward) cover of C
- redundant if, given C_1, C_2, D, if there exists a refinement chain from C_1 to D, it is not through C_2, and if there exists a refinement chain from C_2 to D, it is not through C_1
- proper iff for all C and D if $D \in \rho(C)$ ($D \in \delta(C)$) then $C \not\equiv D$

In the literature [32] two combinations of above properties are currently considered as remarkable characteristics of refinements:

Definition 4.7 (ideality). A refinement operator is *ideal* if it is locally finite, proper and complete.

Definition 4.8 (optimality). A refinement operator is *optimal* if it is locally finite, non-redundant and weakly complete.

Badea and Stanciu in [5] suggest that in hypothesis spaces with rare solutions optimal refinement operators eliminate a source of complexity (redundancy); in fact, it is very likely that the exploration of a huge portion of the whole space is required. In hypothesis spaces with more dense solutions, properness allows for a faster traversal and density ensures the high likelihood of finding a solution along a refinement path.

Past investigations on refinement operators offer many results in a parallel learning paradigm that is Inductive Logic Programming. In this field, there are very important results that have been summed up in [32] and referenced therein. Many of them can be applied also to Description Logics. In particular, the following propositions[6] can be generally applied to search spaces endowed with a partial ordering relationship.

Proposition 4.1 (Non-ideality, specialization case). *Let $\mathcal{S}$ a space partially ordered by $\preceq$ that is the partial ordering imposed by subsumption*[7] *If there exist in $\mathcal{S}$ C and D_n, $n \geq 1$ such that:*

1. $D_1 \prec D_2 \prec \cdots \prec D_n \prec D_{n+1} \prec \cdots \prec C$
2. $\nexists E \in \mathcal{S}\colon \forall n \geq 1 : D_n \preceq E \equiv C$

then no downward ideal operator exists.

Proposition 4.2 (Non-ideality, generalization case). *Let $\mathcal{S}$ be a space partially ordered by $\preceq$, that is the partial ordering imposed by subsumption. If there exist in $\mathcal{S}$ C and D_n, $n \geq 1$ such that:*

1. $C \equiv D_1 \preceq D_2 \prec \cdots \prec D_n$
2. $\nexists E \in \mathcal{S}\colon \forall n \geq 1 : C \prec E \preceq D_n$

then no upward ideal operator exists.

Moreover, Badea and Nienhuys-Cheng (see Example 7 in [4]) showed, for $\mathcal{ALER}$, that there is a tradeoff between completeness and local finiteness.

Example 4.1 (Completeness vs. local finiteness). Consider the concept A, it admits an infinite number of unordered (w.r.t. subsumption) minimal refinements like: $\{A \sqcap \forall R.E_i,$

[6] see, resp., Lemma 1 and 2 in [32].

[7] $C \preceq D \Leftrightarrow C \sqsubseteq D$.

$i \geq 1\}$ where each E_i can be defined as:

$$E_1 = B$$

$$E_i = \forall R.E_{i-1}$$

Therefore, no complete downward refinement can be locally finite.[8]

The problem becomes how to deal with non finiteness, preserving completeness. Badea and Nienhuys-Cheng, in the work mentioned above, observe that, in situations like the one in Example 4.1, hypotheses can be ordered by their depth so there could be some heuristics for hypothesis selection to take the best and shortest one. Another solution could be using examples for pruning the possible suitable refinements instead of merely using them for evaluating, *a posteriori*, each possible candidate. This kind of approach will be used in the following. Our next step will consist now on devising theoretical operators ensuring at least completeness in order to guarantee the completeness of the resulting inductive inferences.

We need to define the notion of atoms that can be used (added/removed) during the refinement process:

Definition 4.9 (DL-Atoms). $\mathcal{DL}-\mathcal{ATOMS}$ is a set of simple descriptions (recursively) defined as follows:

- $P \in N_C \wedge P \not\equiv \top \Rightarrow P \in \mathcal{DL}-\mathcal{ATOMS}$
- $E = \neg P \wedge P \in N_C \wedge P \not\equiv \bot \Rightarrow E \in \mathcal{DL}-\mathcal{ATOMS}$
- $E = \exists R.\top \wedge R \in N_R \Rightarrow E \in \mathcal{DL}-\mathcal{ATOMS}$
- $E = \forall R.C \wedge R \in N_R \wedge C \in \mathcal{DL}-\mathcal{ATOMS} \Rightarrow E \in \mathcal{DL}-\mathcal{ATOMS}$
- $E = \bigsqcup_{i=1}^{n} E_i \wedge E_i \in \mathcal{DL}-\mathcal{ATOMS}$ [If used in ρ] $\Rightarrow E \in \mathcal{DL}-\mathcal{ATOMS}$ or
- $E = \bigsqcap_{i=1}^{n} E_i \wedge E_i \in \mathcal{DL}-\mathcal{ATOMS}$ [If used in δ] $\Rightarrow E \in \mathcal{DL}-\mathcal{ATOMS}$

The theoretical operators are now defined as follows:

Definition 4.10 (Specialization in $\mathcal{ALC}$). Let C be a consistent ($C \not\equiv \bot$) $\mathcal{ALC}$ concept and consider it, without loss of generality, in $\mathcal{ALC}$-normal form. Let ρ be a downward refinement operator such that: $\rho = (\rho_\sqcup, \rho_\sqcap)$ where $\rho_\sqcup, \rho_\sqcap$ are in their turn downward refinements defined as follows:

If $C = \bigsqcup_{i=1}^{n} = C_1 \sqcup C_2 \sqcup \ldots C_n$, with $n \geq 2$, then $\rho(C) = \rho_\sqcup(C)$ else ($n \leq 2$) $\rho(C) = \rho_\sqcap(C)$ ($C = \bigsqcap_{i=1}^{n} D_i$)

where $\rho_\sqcup$ has the following (non deterministic) behavior:

- If $C' = \bigsqcup_{i=1}^{n} C_i \sqcap \bigsqcup_{i=1,\ldots,n}^{i\neq j} C_i = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_{j-1} \sqcup C_{j+1} \sqcup \cdots \sqcup C_n$ with $1 \leq j \leq n$ then $C' \in \rho_\sqcup(C)$ (Remove disjunct)
- If $C'_j \in \rho_\sqcap(C_j)$ then $C' \sqcup C'_j \in \rho_\sqcup(C)$ (Specialize disjunct)

and $\rho_\sqcap$ has the following (non deterministic) behavior:

- If $C' = C \sqcap \textsf{DL-ATOM}$ then $C' \in \rho_\sqcap(C)$ (Add conjunct)
- If $C' = \left(\bigsqcap_{i=1}^{n} D_i \sqcup \bigsqcap_{i=1,\ldots,n}^{i\neq j} D_i\right) \sqcap D'_j = D_1 \sqcap D_2 \sqcap \cdots \sqcap D_{j-1} \sqcap D'_j \sqcap D_{j+1} \sqcap \cdots \sqcap D_n$ with $1 \leq j \leq n$ and $D'_j \in \rho_\sqcup(D_j)$ then $C' \in \rho_\sqcap(C)$ (Specialize conjunct)

Let us now try to verify which properties hold for this specializing operator.

Proposition 4.3 (local finiteness of $\boldsymbol{\rho}$). *ρ in Def. 4.10 is not locally finite.*

Proof: see Appendix. □

Proposition 4.4 (weak completeness of $\boldsymbol{\rho}$). *ρ in Def. 4.10 is weakly complete.*

Proof: see Appendix. □

This result is useful for showing that the following proposition holds:

Proposition 4.5 (completeness of $\boldsymbol{\rho}$). *ρ in Def. 4.10 is complete.*

Proof: see Appendix. □

We now turn to examine the dual refinement operator, i.e. the generalizing operator δ.

Definition 4.11 (Generalization in $\mathcal{ALC}$). Let C be a consistent $\mathcal{ALC}$ concept ($C \not\equiv \top$) and consider it, without loss of generality, in $\mathcal{ALC}$-normal form. Let δ be an upward refinement operator such that: $\delta = (\delta_\sqcup, \delta_\sqcap)$ where $\delta_\sqcup, \delta_\sqcap$ are in their turn upward refinement operators, defined as follows:

$\delta_\sqcup(C)$ (with $C = \bigsqcup_{i=1}^{n} C_i,\ n \geq 1$)

- $C \sqcup \textsf{DL-ATOM} \in \delta_\sqcup(C)$ (Add disjunct) if $\textsf{DL-ATOM} \in \mathcal{DL}-\mathcal{ATOMS} \wedge \textsf{DL-ATOM} \not\sqsubseteq C$
- $C \sqcup \bigsqcup_{i=1,\ldots,n}^{i\neq j} C_i \sqcup C'_j \in \delta_\sqcup(C_j)$ if $1 \leq j \leq n \wedge C'_j \in \delta_\sqcap(C)$ (Generalize disjunct)

where $\delta_\sqcap(C)$ $\left(C = \bigsqcap_{i=1}^{m} C_i,\ m \geq 1\right)$

- $C \sqcup \bigsqcap_{i=1,\ldots,m}^{i\neq j} C_i \in \delta_\sqcap(C)$ if $1 \leq j \leq m$ (Drop conjunct)
- $(C \sqcup \bigsqcap_{i\neq j}{}_{i=1}^{m} C_i) \sqcap C'_j \in \delta_\sqcap(C)$ if $C'_j \in \delta_\sqcup(C_j)$ (Generalize conjunct)

[8] The same holds for generalization as shown in Example 4.2.

We already know (see Proposition 4.2) that there does not exists an ideal δ (according to Def. 4.6). However, an investigation on the degree of completeness of this δ is opportune.

Proposition 4.6 (weak completeness of $\boldsymbol{\delta}$). *δ in Def. 4.11 is weakly complete.*

Proof: see Appendix. □

Proposition 4.7 (completeness of $\boldsymbol{\delta}$). *δ in Def. 4.11 is complete.*

Proof: see Appendix. □

Example 4.2 (local infiniteness of δ). Consider the concept $C = \bot$ and the infinite set of concepts D_i defined as follows for $i \geq 0$ and $R \in N_R$:

$D_0 = \exists R.C$

$D_i = \exists R.D_{i-1}$

all D_i are incomparable therefore a complete upward refinement operator cannot be locally finite.

Furthermore, consider the following infinite descending chain of concepts E_j (in terms of subsumption), for $j \geq 0$:

$E_0 = D_0$

$E_j = D_i \sqcap D_{j-1}$

that together with concept C match the premises of Proposition 4.2.

We have defined two operators that are complete but not locally finite and also highly redundant; to have an idea of the redundancy, consider the example below:

Example 4.3 (redundancy of ρ). Consider the concepts $\forall R.A$ and $\forall R.(A \sqcap B)$. We obtain the latter from the former by means of:

– $\rho_\sqcap$ (Add conjunct) adding $\forall R.B$ as a conjunct that, after normalization, amounts to $\forall R.(A \sqcap B)$ or
– $\rho_\sqcup$ chained to $\rho_\sqcap$ applied to the former, by refining the filler of the value restriction $\forall R.A$

Note that many different chains can be obtained if we consider the generic $\forall R.(\bigsqcap_{i=1}^{k})A_i$ to be refined into $\forall R.(\bigsqcap_{i=1}^{n})A_i$ with $1 \leq i < k \leq n$.

Therefore, the search in such a tangled and infinite space needs pruning strategies that achieve better efficiency that the mere *generate-and-test* strategy. That is where and why examples come into play in the choice, or, better, in the construction of the candidate hypothesis, as we will see in the following definitions.

Definition 4.12 (Example-biased ρ^E). Let $E = E^+ \cup E^-$ be a set of concept descriptions, C be a consistent ($C \not\equiv \bot$) $\mathcal{ALC}$ concept and consider it without loss of generality in $\mathcal{ALC}$-normal form. Let us denote $\#\mathsf{covers(C,E)} = |\{D \in E\colon\ D \sqsubseteq C\}|$ and with ρ a generic downward refinement operator. Let ρ^E be a downward refinement operator such that: $\rho^E = (\rho^E_\sqcup, \rho^E_\sqcap)$ where $\rho^E_\sqcup$ and $\rho^E_\sqcap$ are, in their turn, downward refinements defined as follows:

If $C = \bigsqcup_{i=1}^{n} C_i = C_1 \sqcup C_2 \sqcup \ldots C_n,\ n \geq 2$ then $\rho^E(C) = \rho^E_\sqcup(C)$

else

$(n \leq 2)$ $\rho^E(C) = \rho^E_\sqcap C$ $\left(C = \bigsqcap_{i=1}^{n} D_i\right)$.

Where $\rho^E_\sqcup$ has the following (non deterministic) behavior:

– If $C' = \bigsqcup_{i=1}^{n} C_i \sqcap \bigsqcup_{i=1,\ldots,n}^{i \neq j} C_i = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_{j-1} \sqcup C_{j+1} \sqcup \cdots \sqcup C_n$ with $1 \leq j \leq n$ and $\#\mathsf{covers}(C, E^+) = \#\mathsf{covers}\,(C', E^+) \wedge \#\mathsf{covers}\,(C, E^-) < \#\mathsf{covers}\,(C', E^-)$ then $C' \in \rho^E_\sqcup(C)$
(Remove disjunct)
– If $C'_j \in \rho(C_j)$ and $\#\mathsf{covers}(C, E^+) = \#\mathsf{covers}\,(C' \sqcup C'_j, E^+) \wedge \#\mathsf{covers}\,(C, E^-) < \#\mathsf{covers}\,(C' \sqcup C'_j, E^-)$ then $C' \sqcup C'_j \in \rho^E_\sqcup(C)$
(Specialize disjunct)

and $\rho^E_\sqcap$ has the following (non deterministic) behavior:

– If $C' = C \sqcap \mathsf{DL\text{-}ATOM}$ ($\mathsf{DL\text{-}ATOM} \in \mathcal{DL}-\mathcal{ATOMS}$) and $\#\mathsf{covers}\,(C, E^+) = \#\mathsf{covers}\,(C', E^+) \wedge \#\mathsf{covers}\,(C, E^-) < \#\mathsf{covers}\,(C', E^-)$ then $C' \in \rho^E_\sqcap(C)$
(Add conjunct)
– If $C' = \left(\bigsqcap_{i=1}^{n} D_i \sqcup \bigsqcap_{i=1,\ldots,n}^{i \neq j} D_i\right) \sqcap D'_j = D_1 \sqcap D_2 \sqcap \cdots \sqcap D_{j-1} \sqcap D'_j \sqcap D_{j+1} \sqcap \cdots \sqcap D_n$ with $1 \leq j \leq n \wedge D'_j \in \rho(D_j)$ and $\#\mathsf{covers}\,(C, E^+) = \#\mathsf{covers}\ (C', E^+)$ and $\#\mathsf{covers}\,(C, E^-) < \#\mathsf{covers}\,(C', E^-)$ then $C' \in \rho^E_\sqcap(C)$
(Specialize conjunct)

Definition 4.13 (Example-biased δ^E). Let $E = E^+ \cup E^-$ be a set of concept descriptions, C be a consistent $\mathcal{ALC}$ concept ($C \not\equiv \bot$) and consider it without loss of generality in $\mathcal{ALC}$-normal form, and let us denote $\#\mathsf{covers}\,(C, E) = |\{D \in E\colon\ D \sqsubseteq C\}|$ and with δ a generic upward refinement operator. Let δ^E be an upward refinement operator such that: $\delta^E = (\delta^E_\sqcup, \delta^E_\sqcap)$ where $\delta^E_\sqcup$ and $\delta^E_\sqcap$ are, in their turn, upward refinements defined as follows:

$$\delta^E_\sqcup(C)\left(C = \bigsqcup_{i=1}^{n} C_i, n \geq 1\right)$$

– $C \sqcup \mathsf{DL\text{-}ATOM} \in \delta^E_\sqcup(C)$ if $\mathsf{DL\text{-}ATOM} \in \mathcal{DL}-\mathcal{ATOMS}$, $\mathsf{DL\text{-}ATOM} \not\sqsubseteq C$ and $\#\mathsf{covers}\,(C, E^-) = \#\mathsf{covers}\,(C \sqcup \mathsf{DL\text{-}ATOM}, E^+) \wedge \#\mathsf{covers}\,(C, E^+) < \#\mathsf{covers}\,(C \sqcup \mathsf{DL\text{-}\ ATOM}, E^+)$
(Add disjunct)

- $D = C \sqcup \bigsqcup_{i=1,\ldots,n}^{i \neq j} C_i \sqcup C'_j \in \delta_{\sqcup}^{E}(C)$ with $1 \leq j \leq n$ if $C'_j \in \delta(C_j)$ and $\#\text{covers}(C, E^-) = \#\text{covers}(D, E^+) \wedge \#\text{covers}(C, E^+) < \#\text{covers}(D, E^+)$
  (Generalize disjunct)

  where $\delta_{\sqcap}^{E}(C)$ $\left(C = \bigsqcap_{i=1}^{m} C_i,\ m \geq 1\right)$
- $C' = C \sqcup \bigsqcap_{i=1,\ldots,m}^{i \neq j} C_i \in \delta_{\sqcap}^{E}(C)$ if $1 \leq j \leq m$ and $\#\text{covers}(C, E^-) = \#\text{covers}(C', E^+) \wedge \#\text{covers}(C, E^+) < \#\text{covers}(C', E^+)$
  (Drop conjunct)
- $C' = (C \sqcup \bigsqcap_{i=1,\ldots,m}^{i \neq j} C_i) \sqcap C'_j \in \delta_{\sqcap}^{E}(C)$ if $C'_j \in \delta(C_j)$ and $\#\text{covers}(C, E^-) = \#\text{covers}(C', E^+) \wedge \#\text{covers}(C, E^+) < \#\text{covers}(C', E^+)$
  (Generalize conjunct)

Such operators are very similar to the theoretical ones presented in Def. 4.11 and in Def. 4.10. The novelty consists in the fact that these operators aim at pruning the possible refinements considering just those that match further criteria (i.e. covering or not covering input examples, and cardinality of the sets of covered examples). Such kind of operators may sacrifice completeness by discarding refinements that cannot help in the search for a solution of the learning problem. For instance, the idea underlying the specialization operator is that, when refining an overly general definition in a consistent way (w.r.t. the negative examples it covers), one needs to *blame* the part of the concept definition that is responsible for covering negative instances and eliminate it. An idea for this *blame assignment* may consist in finding out the residual [30] of the wrong definition w.r.t. the covered negative examples as explained later on. Then, once it is found, the responsible (sub-)concept can be negated (since in $\mathcal{ALC}$ negation is allowed in front of complex concepts) and the intersection between the starting overly general concept and the negated residual can be computed (through $\rho_{\sqcap}^{E}$). Obviously, this is a downward refinement, since in set theory we have that if A and B are two sets then $A \cap B \subseteq A$; take $A = C^{\mathcal{I}}$ and $B = (\neg D)^{\mathcal{I}}$, where C is the concept to be refined and D is the residual computed, then we have $C \sqsupset C \sqcap \neg D$. The negated residual is called *counterfactual* [33] and can be generalized in order to eliminate as many negatives as possible from those covered by the inconsistent definition to be refined as specified in the following subsection.

5 Learning concept descriptions

The learning process can start when there are examples and counterexamples in the A-box of a concept for which a new definition is required. The classification of the examples is assumed to be given by a trainer (the domain expert/the knowledge engineer). However, the methodology would apply also for a similar (yet different) setting, where a definition for the target concept is available in a given T-Box, but it turns out to be incorrect (overly general) because it entails some (new) assertions that have been classified as negative for the target concept.

After showing how to lift examples to the concept language level, and the related notion of coverage, we present the learning algorithm. It is essentially based on the specialization of starting hypotheses. The refinement process is based on appending concept descriptions by means of the operators presented in the following. Finally, we present an example to illustrate the whole process.

5.1 Lifting examples to the concept language level

However, in most of the proposed approaches learning these representations the algorithms do not start directly from the assertions in the A-Box. Indeed they rather work by refining concept descriptions; therefore, a possible solution is to make a starting hypothesis out of a single example (or some examples) and then refine it, thus *lifting* assertions to the concept description language beforehand.

The idea, then, is to represent individuals (instances) at the concept level in a similar way to what Cohen and Hirsh suggest in [9]. They fix an integer k as the maximum length of nested restrictions on roles, starting from the concepts an individual belongs to, and progressively adding at most k nested restrictions on the roles (by recursively applying the algorithm to the role fillers).

Example 5.1 (k-bound Abstraction). Let us consider the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with the T-Box $\mathcal{T} = \{A \sqsubseteq \top, B \sqsubseteq \top, C \sqsubseteq \top, \exists R.\top \sqsubseteq \top\}$ and A-box
$\mathcal{A} = \{A(a), B(b), C(c), R(a, b), R(b, c), R(c, a)\}$. Let us set $k = 2$ and compute D as the k-bound concept level approximation of a.
Note that a belongs to the extension of A ($k = 0$).
a takes part in role R with b as a filler and b belongs to the extension of B ($k = 1$).
b takes part to role R with c as a filler and c belongs to the extension of C ($k = 2$).
Therefore $D \equiv A \sqcap \forall R.(B \sqcap \forall R.C)$ according to what was suggested in [9].

The example above suggests two observations:

1. What if we asked for a $k \geq 3$ deep abstraction (notice that next step would have been considering $R(c, a)$)? It would have examined again a, thus cycling on the same assertions. Therefore the bound k ensures not to fall in infinite loops that might easily occur even in tiny A-Boxes as the one in the example.

2. Unfortunately, if we asked a reasoner whether a actually belongs to the computed concept D, it would answer with a discouraging `"no"`. This is due to a common feature adopted by the DLs reasoners: the *Open World Assumption* (OWA).

Differently from the Closed World Assumption (CWA) that is adopted in Logic Programming and database systems, DL reasoners are allowed to infer the truth of universal (or numeric) restrictions ($\forall R.C$) only if this is directly derivable from the knowledge base, and not solely on the ground of the assertions in the A-Box. Indeed, it may always be the case that new assertions contradict an erroneous generalization. This is true also when dealing with negated restrictions (anything that is not asserted or is not provable cannot be assumed as false—negation as failure). There are three ways to circumvent this issue:

1. building up the k-bound abstraction using existential restriction instead of universal ones (this ensures that an individual always belongs to its abstraction);
2. building up the k-bound abstraction using universal restrictions and adding required assertions to the knowledge base;
3. employing an epistemic operator [13] for restricting an assertion to what is currently known to the knowledge base.

If we choose the second alternative in the case shown in Example 5.1, we would obtain the same concept D, but the following assertions should be added to $\mathcal{A}$: $\{\forall R.(B \sqcap \forall R.C)(a), \forall R.C(b)\}$ in order to accomplish with the OWA. This recalls a default reasoning setting, in which some unexpressed knowledge, if consistent with pre-existing one, may be added to the knowledge base. Default reasoning makes it possible in general to introduce negations. Indeed, stating $\forall R.C$ amounts at introducing a negation - recall that $\forall R.C \equiv \neg\exists R.\neg C$.

After such considerations, we can illustrate the notion of example coverage used henceforth in this work: we will say that an hypothesis (concept) covers an example if it subsumes the concept level representative of such an example. More formally, we first define a function called lift that, given an A-Box and an individual, computes the concept level representative of that individual:

Definition 5.1 (Lift function). Given a a space of concepts $\mathcal{S}$ and a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ we define the function $\mathsf{lift}_{\mathcal{A}}\colon \mathsf{Ind}(\mathcal{A}) \mapsto \mathcal{S}$ a *lift function* iff $\forall a \in \mathsf{Ind}(\mathcal{A})\colon C \equiv \mathsf{lift}_{\mathcal{A}(a)} \Rightarrow \mathcal{A} \models_{\mathcal{T}} C(a)$.

Now we can formally define our notion of coverage:

Definition 5.2 (Coverage). Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and a lift function $\mathsf{lift}_{\mathcal{A}}$, we say that a concept C *covers* an individual (example) $a \in \mathsf{Ind}(\mathcal{A})$ iff $\mathsf{lift}_{\mathcal{A}}(a) \sqsubseteq_{\mathcal{T}} C$.

Before illustrating our algorithm, there is an observation about the solutions of a well-posed learning problem (i.e. one admitting at least one solution).

Proposition 5.1. *Consider a learning problem where $E = E^+ \cup E^-$ is the set of examples and counterexamples of the concept C to be learnt. Let $E_C = E_C^+ \cup E_C^-$ be the set obtained by calculating concept level representatives of the corresponding instances, respectively, in E^+ and E^-,*[9] *with $E_C^+ \cap E_C^- = \emptyset$. Let $\mathcal{H}_S$ be the (possibly infinite) set of hypotheses solving the learning problem. Then, it holds that:*

$$\mathsf{lcs}(E_i^+) \in \mathcal{H}_S \vee \mathcal{H}_S = \emptyset$$

where E_i^+ are all the elements of E^+ and the lcs operator returns the least concept subsuming its arguments [2].

Proof: see Appendix. □

With this result we proved that the lcs represents a solution for every well-posed learning problem with the straightforward corollary of a more precise characterization of such kind of learning problems:

Corollary 5.1 (Well-posed learning problem characterization). *Let $\mathcal{S}$ be the set of all possible concept descriptions in a given DL. Given a learning problem (see Def. 4.1) and a lift function $\mathsf{lift}_{\mathcal{A}} : \mathsf{Ind}(\mathcal{A}) \to \mathcal{S}$ for the a A-Box $\mathcal{A}$, suppose to build the set $E_C = \{C \in \mathcal{S} \mid C = \mathsf{lift}_{\mathcal{A}}(e),\ e \in E\}$, dividing it in positive representatives set (E_C^+) and negative ones (E_C^-). The problem is solvable if and only if at least $\mathsf{lcs}(E_C^+)$ is a solution.*

Proof: Obvious for Proposition 5.1. □

However, notice that the existence of a solution depends on the function $\mathsf{lift}_{\mathcal{A}}$. The following example illustrates the importance of this remark.

Example 5.2 (Granularity shift). Consider the T-Box $\mathcal{T} = \{A \sqsubseteq \top,\ \ D \sqsubseteq \top,\ \ E \sqsubseteq \top\}$ and the A-Box $\mathcal{A} = \{R(a,c), E(c), R(b,d), A(a), A(b), D(d)\}$. Suppose we want to learn the concept C from the following assertions on its examples and counterexamples $\mathcal{A}_C = \{C(a), \neg C(b)\}$.

[9] Depending on the chosen strategy it could happen that $|E_C^*| < |E^*|$, $E^* \in \{E^+, E^-\}$, i.e. there is a representative for more than one instance.

Algorithm 1 Assessment of the least k that grants for the existence of a solution

```
$k \leftarrow 1$
$E'_C \leftarrow \{\bot\}$
$E_C = \emptyset$
existsSolution $\leftarrow$ false
while existsSolution = false $\wedge E'_C \neq E_C$ do
  $E'_C \leftarrow E_C$
  $E_C = \emptyset$
  for each $e \in E$ do
    $E_C = E_C \cup \{k-$bound approximation of msc$(e)\}$
  if lcs$(E_C)$ is not a solution then
    existsSolution $\leftarrow$ false
    $k \leftarrow k + 1$
```

Let us choose the first $\mathsf{lift}_{\mathcal{A}}$, say l^1 such that $l^1(e) = \bigsqcap_{F \in \mathsf{realization}(e)} F$ (i.e. the conjunction of most specific atomic concepts in the realization[10] of an individual). Then $l^1(a) = l^1(b) = A$. In this case we have just one positive example and, from the definition of lcs,[11] we have that our positive lcs is A. Unfortunately, it also covers a negative example, hence the learning problem does not have a solution.

If we choose, instead, as $\mathsf{lift}_{\mathcal{A}}$ the function $l^2 = 1$-bound approximation of the msc of each individual we would have:

$$l^2(a) = A \sqcap \exists R.E$$

$$l^2(b) = A \sqcap \exists R.D$$

the problem admits a solution (at least) that is $A \sqcap \exists R.E$

This example leaves an open question. How to determine, then, a suitable $\mathsf{lift}_{\mathcal{A}}$? We present a correct and complete algorithm (Algorithm 1) for individuating the least bound, in terms of the maximum depth of the representatives, among the possibly infinite k-bound msc approximations.

5.2 YinYang

It is now possible to illustrate the learning algorithm, whose name is YINYANG,[12] that is reported as Algorithm 2. It relies on two interleaving routines performing, respectively, generalization and specialization, that call each other to converge to a correct concept definition.

The generalization algorithm is a greedy covering one: it tries to explain the positive examples by constructing a disjunctive definition. At each outer iteration, a very specialized definition (selected from the residual uncovered concept level representative of examples) is selected as a starting seed for a new partial generalization; then, iteratively, the hypothesis is generalized by means of the upward operator δ (here it is left undefined but the implementor should preferably choose one with a heuristic that privileges the refinements that cover the biggest set of the positives) until all positive concept representatives are covered or some negative representatives are covered. In such a case, the current concept definition *ParGen* has to be specialized. Differently from previous works [14, 15, 22] in which YINYANG had only one strategy to specialize based on counterfactuals, we endowed it with another one implementing basically the theoretical Add conjunct $\rho_{\sqcap}^{E}$ (see Def. 4.12). Below we describe each of the specialization strategies.

Specializing by means of counterfactuals. A counterfactual is a generalization of an exception. This intuition is due to Vere [33], who formalized a previous idea by Winston for eliminating the negative examples covered by an overly general hypothesis generated during the learning process. The specialize co-routine is a transposition of Vere's counterfactuals in DL. It receives the covered examples as its input, finds a sub-description K that is capable of ruling out the negative examples previously covered.

In the routine for building counterfactuals, given a previously computed hypothesis *ParGen*, which is supposed to be complete (covering the positive assertions) yet inconsistent with respect to some negative assertions, the aim is finding those counterfactuals to be added through a conjunction to the initial hypothesis to restore its correctness, ruling out the previously covered negative instances.

The algorithm is based on the construction of residual learning problems based on the sub-descriptions that caused the subsumption of the negative examples, represented by their msc's. In this case, for each model a residual[13] is derived by considering that part of the incorrect definition *ParGen* that did not play a role in the subsumption. The residual will be successively employed as a positive instance of that part of description that should be ruled out of the definition (through negation). Analogously, positive example representatives will play the opposite role of

[10] The realization of an individual is the set of all the most specific atomic concepts the individual is instance of [2].

[11] The lcs of a single concept is the concept itself.

[12] Standing for *Yet another INduction Yields to ANother Generalization*.

[13] Residuals are computed according to Teege's definition [30].

Algorithm 2 YINYANG algorithm

```
Generalize(Positives, Negatives)
input: Positives, Negatives: positive and negative msc approximations
output: Generalization: generalized concept definition
ResPositives ← Positives
Generalization ← $\bot$
while ResPositives $\neq \emptyset$ do
   $ParGen$ ← select_seed$_M$(ResPositives)
   CoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
   CoveredNeg ← $\{Neg \in$ Negatives $\mid ParGen \sqsupseteq Neg\}$
   while CoveredPos $\neq$ ResPositives $\wedge$ $\delta$ returns a proper refinement $\wedge$
   CoveredNeg $= \emptyset$ do
      $ParGen$ ← select($\delta(ParGen)$, ResPositives, Negatives)
      CoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
      CoveredNeg ← $\{Neg \in$ Negatives $\mid ParGen \sqsupseteq Neg\}$
   if CoveredNeg $\neq \emptyset$ then
      $K$ ← counterfactual($ParGen$, CoveredPos, CoveredNeg)
      ParGen ← ParGen $\sqcap \neg K$
      CurrentCoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
      LeftOutPositives ← CoveredPos $\setminus$ CurrentCoveredPos
      if |LeftOutPositives| $> 0$ then
         LeftInPositives ← CurrentCoveredPos $\setminus$ CoveredPos
         if LeftInPositives $= \emptyset$ then
            ParGen ← Add conjunct$\rho^E_\sqcap$(ParGen, LeftOutPositives, coveredNegatives)
            if Add conjunct$\rho^E_\sqcap$ failed then
               ParGen ← lcs(coveredPositives)
         else
            CoveredPos ← LeftInPositives
   Generalization ← Generalization $\sqcup$ ParGen
   ResPositives ← ResPositives $\setminus$ CoveredPos
return Generalization

counterfactual(ParGen, CoveredPos, CoveredNeg, $K$)
Input ParGen: inconsistent concept definition
Input CoveredPos, CoveredNeg: covered positive and negative msc approx.
Output $K$: a counterfactual
NewPositives ← $\emptyset$
NewNegatives ← $\emptyset$
for $N_i \in$ CoveredNeg do
   NewP$_i$ ← residual($N_i$, ParGen)
   NewPositives ← NewPositives $\cup$ {NewP$_i$}
for $P_j \in$ CoveredPos do
   NewN$_j$ ← residual($P_j$, ParGen)
   NewNegatives ← NewNegatives $\cup$ {NewN$_j$}
$K$ ← generalize(NewPositives, NewNegatives)
return $K$
```

negative instances for the residual learning problem under construction.

Finally, this problem is solved by calling the co-routine which generalizes these example descriptions and then conjoining its negation of the returned result.

Specialization by means of add conjunct $\rho^E_\sqcap$. The necessity of providing another specialization strategy besides the counterfactuals is illustrated the following example.

Example 5.3. (Incorrectness of counterfactual-based specialization) The specialization that exploits counterfactuals is not correct. Supposing it as correct, then it would return a correct solution for the following specialization problem:

ParGen $= A$
and
Positives $= \{A \sqcap \exists R.A,\ A \sqcap \exists R.B\}$
Negatives $= \{A \sqcap \exists R.C\}$

It covers the single element of Negatives. Suppose to compute residuals using the Structural difference (see [30]). We have:

NewNeg$_1 = \exists R.A$
NewNeg$_2 = \exists R.B$
NewPos$_1 = \exists R.C$

The inner call to Generalize would return $\exists R.C$. Then our $K = \exists R.C$. Then, eventually, the returned Generalization will be $A \sqcap \neg\exists R.C$ or pushing in depth negation:

Algorithm 3 Add Conjunct $\rho^E_\sqcap$ Implementation

```
$\rho^E_\sqcap$(c, coveredPositives, negatives)
Input: c is a Concept
Input: coveredPositives is a set of positive examples concept level representatives
Input: negatives is a set of negative examples concept level representatives
Output: A specialization covering all elements within input when possible or c itself
toReturnConcept ← c
noPossibleRefinementKeepingAllPositives ← false
moreChanches ← true
cs ← $\emptyset$
alreadyChosenConjuncts ← new empty map that associates positive examples with already used conjuncts
coveredNegatives ← negatives
while |coveredNegatives| > 0 $\wedge$ ($\neg$noPossibleRefinementKeepingAllPositives $\vee$ moreChanches) do
  moreChanches ← false
  for each p $\in$ coveredPositives do
    possiblyLeftOut ← $\emptyset$
    if $p \not\sqsubseteq \bigsqcup_{C \in \mathsf{cs}} C$ then
      availableConjuncts ← chooseConjunct
              ($p$, toReturnConcept, coveredNegatives, alreadyChosenConjuncts)
      aSelectedConjunct ← selectConjunct(availableConjuncts)
      if $lVert$availableConjuncts| > 1 then
        moreChanches ← true
      if aSelectedConjunct $\not\sqsubseteq \bot$ then
        cs ← cs $\cup$ {aSelectedConjunct}
        Add aSelectedConjunct to alreadyChosenConjuncts for p
  if |cs| > 0 then
    toReturnConcept ← $c \sqcap$ lcs(cs)
    noPossibleRefinementKeepingAllPositives
                 ← $\neg(\forall p \in$ coveredPositives : $p \sqsubseteq$ lcs(cs))
    if noPossibleRefinementKeepingAllPositives then
      cs ← $\emptyset$
  else
    noPossibleRefinementKeepingAllPositives ← true
  coveredNegatives ← {$n \in$ negatives, $n \sqsubseteq$ toReturnConcept}
return toReturnConcept
```

$A \sqcap \forall R.(\neg C)$. This generalization does not subsume any of the positives so it is not a correct solution.

One possible objection is that structural residuals is not the actual residual in $\mathcal{ALC}$. Let us try with non-structural but actual residuals then:

$$\mathsf{NewNeg}_1 = (A \sqcap \exists R.A) \sqcup \neg A \equiv \exists R.A \sqcup \neg A$$
$$\mathsf{NewNeg}_2 = (A \sqcap \exists R.B) \sqcup \neg A \equiv \exists R.B \sqcup \neg A$$
$$\mathsf{NewPos}_1 = (A \sqcap \exists R.C) \sqcup \neg A \equiv \exists R.C \sqcup \neg A$$

The inner call to Generalize would return $\exists R.C \sqcup \neg A$. Then $K = \exists R.C \sqcup \neg A$ Eventually, the returned Generalization will be $A \sqcap \neg(\exists R.C \sqcup \neg A) \equiv A \sqcap \forall R.C$. Unfortunately this is the same incorrect solution.

Therefore, there are specialization problems that cannot be solved using counterfactuals and intuitively this happens when counterfactuals involve value restrictions that do not appear in positive examples and cannot be inferred in settings adopting the OWA, such as DLs.[14] Algorithm 5.2 shows an alternative specialization strategy used, as in Algorithm 2, whenever the specialization performed by means of counterfactuals completely fails.[15] Add Conjunct $\rho^E_\sqcap$ consists in adding a new conjunct to the overly general partial generalization ParGen. The problem is that we need to add a conjunct that has all the following features:

- it does not already occur in/does not subsume ParGen
- it does not already occur in/does not subsume any of the already covered negatives
- it subsumes each of the covered positives

The idea is that every positive contributes with at most one conjunct. The logic of choosing a conjunct is the base for the chooseConjunct subroutine. For each positive example covered by ParGen it selects one of its conjuncts fulfilling the criteria above. However, if some previously examined

[14] Notice that, in Logic Programming, Negation As Failure and Closed World Assumption make counterfactuals always correct.

[15] Notice that partial failures mean that counterfactuals ruled out only part of the formerly covered positives. Such positives are simply put back in residual positives set and Add Conjunct $\rho^E_\sqcap$ is **not** invoked.

positive provided a conjunct that subsumes it, the current positive is skipped as useless for providing conjuncts. After all the positives have been examined, all the conjuncts that they issued are aggregated in a disjunction (in other words the lcs of all the conjuncts is computed) and such disjunction is conjoined to ParGen. In this way we obtain a specialization that still covers all the positives but none of the formerly covered negatives. Notice that there are cases in which neither a positive can provide a conjunct on its own that matches the above criteria nor there is another positive providing a suitable one covering it. These are cases in which the specialization fails and YinYang replaces an overly general ParGen with the lcs of the formerly covered positives closing the inner loop and starts another outer loop in Algorithm 2, with the selection of another seed from the remaining positive examples, if any. Furthermore, in order to ensure completeness, Add conjunct $\rho_{\sqcap}^{E}$ should consider positives in any possible order, and, if a positive could provide more than one conjunct suitable for specialization, it should take into account each of them. Implementing this strategy yields a dramatic increase of the computational complexity that proved itself practically unfeasible even for small sets of positives. We preferred a fixed order that, though possibly leading to incompleteness, makes the method tractable.

5.3 Running an example

In this section we present a short example in order to illustrate the algorithm through its trace.

Example 5.4 (Supervised Learning). Suppose that the starting knowledge base is made up of

$$\mathcal{T} = \{A \sqsubset \top, B \sqsubset \top, C \sqsubset \top, D \sqsubset \top\}$$

$$\begin{aligned}\mathcal{A} = \{&A(a_1), (\forall R.(\neg A \sqcap B \sqcap C))(a_1), A(a_2),\\ &(\forall R.(\neg A \sqcap B \sqcap D))(a_2), A(a_3),\\ &R(a_3, b), A(a_4), R(a_4, a_5), A(a_5),\\ &(\neg B)(a_5), A(b), B(b)\}\end{aligned}$$

Let C_L be the target concept to learn with this related training set of assertions:
$\mathcal{A}_{C_L} = \{C_L(a_1), C_L(a_2), (\neg C_L)(a_3), (\neg C_L)(a_4)\}$.

Thus the examples and counterexamples are, respectively: Positives $= \{a_1, a_2\}$ and Negatives $= \{a_3, a_4\}$.

The approximated concept level representatives[16] are:

[16] Here msc* stands for the approximation of the most specific concept operator, a well-known non standard inference in DLs [3].

$\mathsf{msc}^*(a_1) = A \sqcap \exists R.(\neg A \sqcap B \sqcap C)$
$\mathsf{msc}^*(a_2) = A \sqcap \exists R.(\neg A \sqcap B \sqcap D)$
$\mathsf{msc}^*(a_3) = A \sqcap \exists R.(A \sqcap B)$
$\mathsf{msc}^*(a_4) = A \sqcap \exists R.(A \sqcap \neg B)$

The trace of the algorithm in this case follows:

```
generalize:
ResidualPositives ← {msc*(a_1), msc*(a_2)}
Generalization ← ⊥
    /* Outer while loop */
    ParGen ← $\mathsf{msc}^*(a_1) = A \sqcap \exists R.(\neg A \sqcap B \sqcap C)$
    CoveredPos ← {msc*(a_1)}
    CoveredNeg ← {}
    ParGen ← $A$        /* restriction dropped in the inner
        loop */
    CoveredPos ← {msc*(a_1), msc*(a_2)}
    CoveredNeg ← {msc*(a_3), msc*(a_4)}
    Call counterfactuals ($A$, {msc*(a_1), msc*(a_2)},
        {msc*(a_3), msc*(a_4)})

    counterfactuals:
        NewP_1 ← $\exists R.(A \sqcap B)$
        NewPositives ← $\{\exists R.(A \sqcap B)\}$
        NewP_2 ← $\exists R.(A \sqcap \neg B)$
        NewPositives ← $\{\exists R.(A \sqcap B), \exists R.(A \sqcap \neg B)\}$
        NewN_1 ←= $\exists R.(\neg A \sqcap B \sqcap C)$
        NewNegatives ← $\{\exists R.(\neg A \sqcap B \sqcap C)\}$
        NewN_2 ←= $\exists R.(\neg A \sqcap B \sqcap D)$
        NewNegatives ← $\{\exists R.(\neg A \sqcap B \sqcap C), \exists R.$
            $(\neg A \sqcap B \sqcap D)\}$
        Call generalize ($\{\exists R.(A \sqcap B), \exists R.(A \sqcap \neg B)\}$,
        $\{\exists R.(\neg A \sqcap B \sqcap C), \exists R.(\neg A \sqcap B \sqcap D)\}$)
        ⋮
```

Returning from the recursion to the main generalization results in $C_L = A \sqcap \exists R.(\neg A)$.

Notice that, in the example above, we did not detail the generalization step as there is more than one possibility as it will be shown in the empirical evaluation. The residual calculation has been performed using what Teege calls RCFs [30]. Indeed, though in $\mathcal{ALC}$ the actual residual is not calculated by means of the RCFs, it can be shown that the actual residual always subsumes the one computed by RCFs (straightforward from residual definition). Furthermore, as Teege himself argues, an RCFs based residual is more informative than the actual residual in individuating the parts of a definition which are responsible for the coverage of a concept w.r.t. one another. This is true, especially for hypotheses and examples that, as in our case, are in conjunctive form.

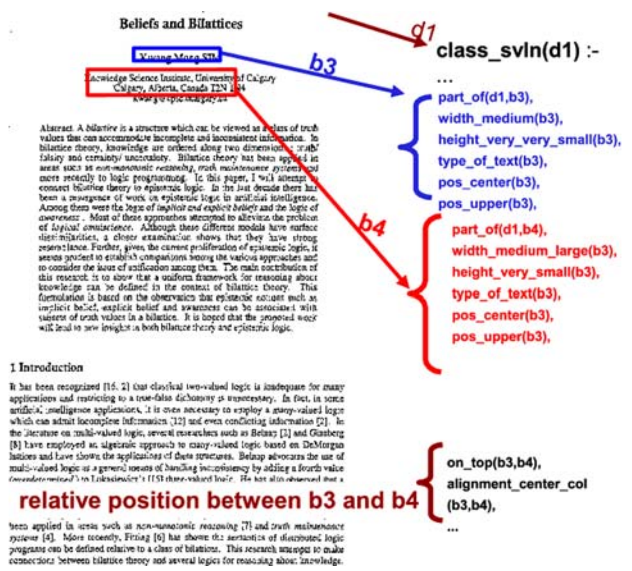

Fig. 1 Example of a document logical description

6 Evaluation

6.1 Dataset descriptions

The datasets used for this empirical evaluation are 15. They have already been used for evaluating the ILP system INTHELEX, with the results are reported in [17]. Such datasets where automatically generated by WISDOM++ [16] starting from a set of 90 scientific papers submitted to two international conferences and a journal, namely the *International Symposium on Methodologies for Intelligent Systems*, the *International Conference on Machine Learning* and the *Transactions on Pattern Analysis and Machine Intelligence*. Every document is represented as a set of DATALOG assertions such as those sketched in Fig. 1.

Each dataset represents a learning problem. The datasets are partitioned according to the format of the paper as follows:

- SVLN indicates papers submitted to ISMIS Conference.
- ICML indicates papers submitted to the ICML Conference.
- TPAMI indicates the papers submitted to the TPAMI journal.

We encoded in the dataset name the kind of learning problem to be solved, whereas *XXX* is a placeholder for the format, hence:

- *XXX*Astract: the objective is to learn the layout description of a block containing the abstract of a paper;
- *XXX*Affiliation: learn the layout description of a block containing the affiliations of the authors of a paper (it occurs in the TPAMI and SVLN datasets only);
- *XXX*Author: learn the layout description of a block containing the authors of a paper;
- *XXX*Index_term: learn the layout description of a block containing the keywords of a paper (it occurs in the TPAMI dataset only);
- *XXX*Page_num: learn the layout description of a block containing the page numbers of a paper (present in TPAMI and ICML datasets);
- *XXX*Running_head: learn the layout description of a block containing the running heads in the paper (it occurs in the TPAMI dataset only);
- *XXX*Title: learn the layout description of a block containing the title of a paper.

In Fig. 2 we report a description of a positive example of a layout block representing the abstract for an SVLN paper. The right reading for such an example is "*the block named* `ismis10_6` *is an abstract because it is part of block* `ismis10_1` *and there are blocks the* `ismis10_2`*, ismis10_3,... that are part of* `ismis10_1`*...*".[17] This kind of representation of examples (and counterexamples) was introduced by Michalski [25] and is called *complete relevance*. It means that learners will have as an input associated to each (counter)example all the relevant descriptions.

Though very large, all datasets fall into the sub-language of Horn Logic that can be translated into $\mathcal{ALC}$ Description Logic. The only adjustment that was needed is that the asymmetrical nature of `part_of` was used to saturate the examples description in simulating the presence of the background knowledge axiom:[18]

```
part_of(A,B) :- is_part_of(B,A).
```

We performed the transformation of the 15 DATALOG datasets into the $\mathcal{ALC}$ DL[19] and we proceeded setting up the experiments described in the next section.

[17] Predicates like `very_small` and `small` are used to let INTHELEX deal with discredited numeric intervals that in this way become logic predicates.

[18] The reason for this is that inverse roles are not allowed in $\mathcal{ALC}$, hence `part_of` could not be used as their direction in the starting dataset was from the container towards the content, but examples blocks were very often contained by some other block.

[19] Actually, instead of using a proprietary DL formalism, we turned the dataset into OWL-DL ontologies—see `http://www.w3.org/2004/OWL/`.

Fig. 2 Excerpt of a typical example description. It has been cut for readability sake: dots stand for omitted very similar facts w.r.t. the previous ones

```
logic_type_abstract(ismis10_6) :- part_of(ismis10_1, ismis10_2),
part_of(ismis10_1, ismis10_3), part_of(ismis10_1, ismis10_4),
part_of(ismis10_1, ismis10_5), part_of(ismis10_1, ismis10_6),
part_of(ismis10_1, ismis10_7), ... width_medium_small(ismis10_2),
width_large(ismis10_3), ... height_medium_large(ismis10_6),
height_very_very_small(ismis10_7), ... type_of_text(ismis10_2),
type_of_text(ismis10_3), type_of_text(ismis10_4), type_of_text(ismis10_5),
type_of_text(ismis10_6), ... pos_left(ismis10_2), pos_center(ismis10_3),
pos_center(ismis10_4), ... pos_lower(ismis10_2), pos_upper(ismis10_3),
pos_upper(ismis10_4), ... on_top(ismis10_3, ismis10_4), on_top(ismis10_3,
ismis10_5), ... to_right(ismis10_8, ismis10_12), to_right(ismis10_6,
ismis10_11), alignment_left_col(ismis10_3, ismis10_7),
alignment_right_col(ismis10_3, ismis10_8), ... alignment_middle_row(ismis10_6,
ismis10_11).
```

6.2 Empirical evaluation: Results and discussion

The adopted methodology was a 10-fold cross validation. The fold division was totally random for each experiment. 60 experiments have been carried out, in particular:

- 15 experiments performed using YinYang endowed with upward refinement operator that generalizes an hypothesis dropping one conjunct at a time; the results are shown in Table 2;
- 15 experiments performed using YinYang endowed with the same generalization operator as above and with the induction of default knowledge; the results are shown in Table 3;
- 15 experiments carried out using YinYang endowed with the generalization algorithm (greedy generalization) that keeps on dropping conjuncts from an hypothesis to be refined until either no residual positives are covered or some negative is covered; the outcomes are shown in Table 4;
- 15 experiments carried out using an implementation of the algorithm by Cohen and Hirsh [9] for the $\mathcal{ALC}$ DL. The outcomes are shown in Table 5. These represent the baseline for a comparison with the three versions of YinYang.

As shown in the tables and, more clearly, in Figs. 3, 4, 5 and 6, on average, Cohen and Hirsh's algorithm is always outperformed by the three versions of YinYang. This is because such an algorithm uses the lcs as its generalizing operator and this, as expected, often overfits the data.[20]

There are, however, some cases in which the version under comparison of our system performs with a very low recall. This occurs, during the learning process, when the system has to specialize a very general hypothesis (i.e. one covering many negative examples) and the specialization routines (on account of their incompleteness—see the previous section) fail at finding a suitable refinement. In such situations, the overly general hypothesis is discarded and the lcs of the formerly covered positive is taken into consideration. This results in a dramatic reduction of the recall and in an increase of the length of the learned definition.

Default reasoning seems to work much better, though there are some caveats one should take into account when deciding whether to activate or not such option. First of all, performance may be an issue. Although it has not been evaluated thoroughly, working with induced default knowledge means adding some non-trivial complexity to concept level representatives of the examples. This implies an increased reasoning complexity and more verbose definitions. Furthermore, one should keep in mind that, in order to comply with the OWA, the induced default assertions have to be saved into the knowledge base that, hence, will be modified after the learning process and this could be often undesirable.

There are, however, some promising directions of investigation for improving the obtained results. First of all, one could think of the combination of various generalization operators in order to make YinYang more flexible. When for instance, an overly general hypothesis cannot be specialized but by replacing it with the lcs of the covered positives, one could imagine to *backtrack* and choose another generalization operator that computes shorter refinements steps and can return not so general hypotheses, that hopefully cover the majority of the previously covered positives. This could prevent the system from recurring to the lcs and decrease the recall of the final output.

Obviously, there can be other generalization strategies and, among them, it could be extremely interesting to investigate top-down strategies for generalizing instead of the bottom-up ones, like those adopted in the algorithm. Indeed, top down algorithms in the literature are known for producing high recall rates though there is the risk of lowering the precision, depending on the effectiveness of specialization on the generated hypotheses.

Last but not least, there is the possibility of further refining the obtained descriptions after the learning process is

[20] Notice that the lcs in $\mathcal{ALC}$ consists of the mere disjunction of its input descriptions, hence its recall is not that high in general.

Table 2 Results obtained running YinYang with the generalization operator that drops one conjunct per time

Dataset	Train. set size	Precision	Recall	Pred. acc.	F-Measure
SVLN Abstract	32+;250−	89.20%	55.00%	94.20%	68.04%
SVLN Affiliation	30+;252−	86.70%	32.80%	92.50%	47.59%
SVLN Author	30+;252−	25.00%	18.10%	90.00%	21.00%
SVLN Title	30+;252−	60.00%	33.90%	92.50%	43.32%
ICML Abstract	28+;340−	86.70%	78.30%	98.30%	82.29%
ICML Author	36+;332−	95.00%	93.80%	98.90%	94.40%
ICML Page number	27+;341−	100.00%	84.30%	98.60%	91.48%
ICML Title	29+;339−	87.50%	54.17%	94.44%	66.91%
TPAMI Abstract	36+;576−	50.00%	19.00%	95.08%	27.54%
TPAMI Affiliation	31+;581−	80.00%	45.83%	97.04%	58.28%
TPAMI Author	34+;578−	80.00%	62.50%	97.86%	70.18%
TPAMI Index term	32+;580−	50.00%	25.63%	95.90%	33.89%
TPAMI Page number	33+;579−	80.00%	36.00%	96.72%	49.66%
TPAMI Running Hd.	40+;572−	87.50%	30.55%	90.16%	45.29%
TPAMI Title	40+;572−	25.00%	8.30%	90.98%	12.46%
Average	–	72.17%	45.21%	94.88%	54.15%

Table 3 YinYang plus default knowledge induction - experimental results

Dataset	Tr. set size	Precision	Recall	Pred. acc.	F-Measure
SVLN Abstract	32+;250−	100,00%	83,43%	96,43%	90,97%
SVLN Affiliation	30+;252−	100,00%	83,83%	98,21%	91,21%
SVLN Author	30+;252−	91,67%	87,22%	96,43%	89,39%
SVLN Title	30+;252−	90,00%	81,07%	98,21%	85,30%
ICML Abstract	28+;340−	90,00%	90,00%	99,17%	90,00%
ICML Author	36+;332−	100,00%	97,50%	99,72%	98,73%
ICML Page number	27+;341−	100,00%	100,00%	100,00%	100,00%
ICML Title	29+;339−	100,00%	95,00%	99,44%	97,44%
TPAMI Abstract	36+;576−	90,00%	90,00%	99,84%	90,00%
TPAMI Affiliation	31+;581−	100,00%	100,00%	100,00%	100,00%
TPAMI Author	34+;578−	100,00%	93,33%	99,67%	96,55%
TPAMI Index term	32+;580−	100,00%	91,67%	99,18%	95,65%
TPAMI Page number	33+;579−	100,00%	100,00%	100,00%	100,00%
TPAMI Running Hd.	40+;572−	86,67%	89,00%	99,02%	87,82%
TPAMI Title	40+;572−	96,89%	93,33%	99,34%	95,08%
Average	–	96,35%	91,69%	98,98%	93,88%

Table 4 YinYang with greedy generalization - experimental results

Dataset	Tr. set size	Precision	Recall	Pred. acc.	F-Measure
SVLN Abstract	32+;250−	80,00%	39,00%	92,50%	52,44%
SVLN Affiliation	30+;252−	91,67%	84,75%	96,43%	88,07%
SVLN Author	30+;252−	55,00%	23,17%	91,07%	32,60%
SVLN Title	30+;252−	85,00%	50,83%	93,57%	63,62%
ICML Abstract	28+;340−	91,67%	85,00%	97,50%	88,21%
ICML Author	36+;332−	95,00%	71,00%	96,67%	81,27%
ICML Page number	27+;341−	97,50%	84,17%	98,33%	90,35%
ICML Title	29+;339−	77,50%	62,50%	96,67%	69,20%
TPAMI Abstract	36+;576−	70,00%	27,00%	95,74%	38,97%
TPAMI Affiliation	31+;581−	98,57%	89,67%	99,18%	93,91%
TPAMI Author	34+;578−	94,17%	72,08%	97,87%	81,66%
TPAMI Index term	32+;580−	90,00%	55,17%	97,70%	68,41%
TPAMI Page number	33+;579−	98,33%	100,00%	99,84%	99,16%
TPAMI Running Hd.	40+;572−	85,71%	74,64%	97,87%	79,79%
TPAMI Title	40+;572−	43,37%	70,64%	91,48%	53,74%
Average	–	83,57%	65,97%	96,16%	72,09%

Table 5 Cohen e Hirsh algorithm implementation for $\mathcal{ALC}$ results

Dataset	Tr. set size	Precision	Recall	Pred. acc.	F-Measure
SVLN Abstract	32+;250−	70,00%	32,83%	91,42%	44,70%
SVLN Affiliation	30+;252−	60,00%	33,16%	92,85%	42,71%
SVLN Author	30+;252−	40,00%	23,67%	90,71%	29,74%
SVLN Title	30+;252−	30,00%	12,00%	90,35%	17,14%
ICML Abstract	28+;340−	40,00%	12,83%	93,61%	19,43%
ICML Author	36+;332−	90,00%	65,67%	97,22%	75,93%
ICML Page number	27+;341−	90,00%	78,33%	98,61%	83,76%
ICML Title	29+;339−	0,00%	0,00%	99,44%	91,94%
TPAMI Abstract	36+;576−	50,00%	32,50%	95,08%	39,39%
TPAMI Affiliation	31+;581−	100,00%	100,00%	100,00%	100,00%
TPAMI Author	34+;578−	100,00%	93,33%	99,67%	96,55%
TPAMI Index term	32+;580−	70,00%	32,78%	95,90%	44,65%
TPAMI Page number	33+;579−	80,00%	36,00%	96,72%	49,66%
TPAMI Running Hd.	40+;572−	80,00%	41,50%	96,72%	54,65%
TPAMI Title	40+;572−	10,00%	1,42%	93,61%	2,49%
Average	–	59,33%	33,71%	94,64%	42,30%

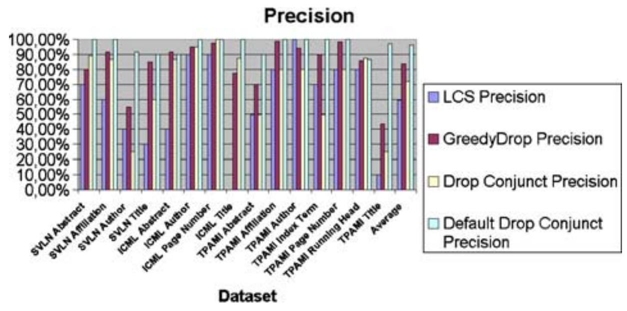

Fig. 3 Result comparisons: Precision

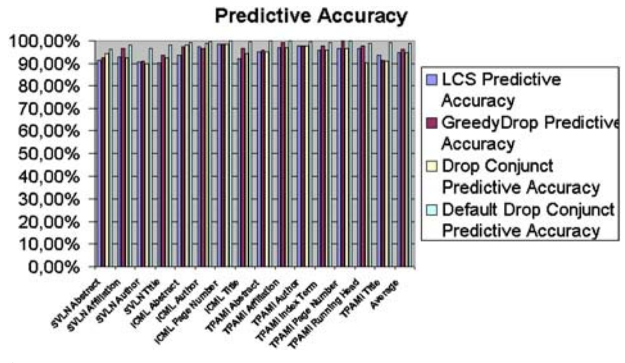

Fig. 5 Result comparisons: Predictive accuracy

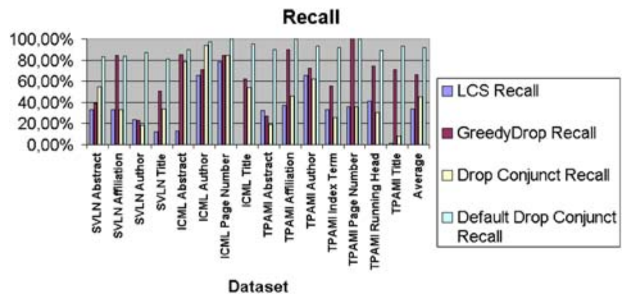

Fig. 4 Result comparisons: Recall

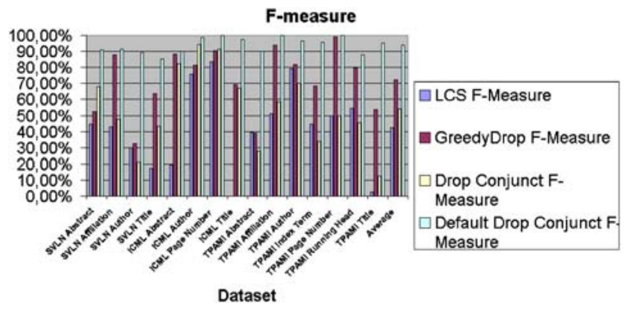

Fig. 6 Result comparisons: F-Measures

done. One can imagine, for instance, that, after the learning phase, the output is a concept definition with many disjuncts. Among them there could be some which turn out to be very *similar*. Then, it could be possible to synthesize them into a unique small generalization of the two, of course if this preserves completeness and consistency of the learned descriptions. For these purposes, *similarity* measures for DLs are necessary such as those lately proposed in the literature [7, 10, 11]; hence, soon we expect to employ them and explore this direction.

7 Conclusion

In this paper we have tackled the problem of constructing ontologies in a semi-automatic fashion. In particular, we have presented an algorithm that is able to infer concept descriptions in the Description Logic $\mathcal{ALC}$ from concept instances available in an A-box. The algorithm adopts a sort of greedy coverage strategy, hence its complexity depends on the complexity of reasoning in the adopted DL language, namely subsumption in $\mathcal{ALC}$ which is P-Space [2].

The algorithm can represent the basis for a powerful tool for knowledge engineers. It has been implemented in a system called YINYANG (Yet another INduction Yields to ANother Generalization).

Experiments have been carried out in order to assess the effectiveness of our approach on a particular domain that is the Document Image Understanding. Results that, though the system behaves better than the standard generalization obtained using the lcs operator, the descriptions readability and the F-Measure may be further improved. This can be accomplished by studying more complex generalization strategies that may dynamically adjust the pace of the refinement, when overly general hypotheses are produced and they cannot be efficiently specialized.

Appendix: Proofs

Proposition 4.3. *ρ in Def. 4.10 is not locally finite.*

Proof: Consider the set $\mathcal{DL}-\mathcal{ATOMS}$. It has non-finite cardinality as:

$$\begin{aligned} E = \forall R.C,\ R \in N_R \wedge C \in \mathcal{DL}-\mathcal{ATOMS} \\ \rightarrow E \in \mathcal{DL}-\mathcal{ATOMS} \end{aligned}$$

This is needed to create the infinite set of incomparable value restriction illustrated before in Example 4.1. Furthermore, consider the following infinite ascending chain of concepts (in terms of subsumption) E_j for $j \geq 0$ and $R \in N_R$:

$$\begin{aligned} D_0 &= \exists R.\top \\ D_i &= \exists R.D_{i-1} \\ E_0 &= D_0 \\ E_j &= D_i \sqcup D_{j-1} \end{aligned}$$

that, together with concept C, match the premises of Proposition 4.1. □

Proposition 4.4. *ρ in Def. 4.10 is weakly complete.*

Proof: Recall from Def. 4.6 that ρ is weakly complete means that $\forall C \in \mathcal{ALC}$: $C \in \rho^*(\top)$. Suppose, without loss of generality, that C is in $\mathcal{ALC}$ normal form, hence it can be written like:

$$C = \bigsqcup_{i=1}^{n} C_i = C_1 \sqcup C_2 \sqcup \ldots \sqcup C_n$$

for some $1 \leq i \leq n$, $n \geq 1$ and C_i a conjunction including atomic concepts, existential restrictions and one value restriction per role (see Def. 3.2). We shall prove that the generic concept C can be obtained starting from $\top$. Let us proceed by induction on the depth of C.

$\mathsf{depth}(C) = 0$

This means that C is in the form:

$$C = \bigsqcup_{i=1}^{n} \bigsqcap_{j_i=1}^{m_i} \mathsf{PONEGP}_{j_i}$$

where $\forall i \in \{1, \ldots n\}$: $m_i \geq 1$ and $\mathsf{PONEGP}_{j_i} \in \{P, \neg P \mid P \in N_P\}$ or, more informally:

$$\begin{aligned} C = & (\mathsf{PONEGP}_{1_1} \sqcap \mathsf{PONEGP}_{2_1} \sqcap \cdots \sqcap \mathsf{PONEGP}_{m_1}) \sqcup \\ & \sqcup (\mathsf{PONEGP}_{1_2} \sqcap \mathsf{PONEGP}_{2_2} \sqcap \cdots \sqcap \mathsf{PONEGP}_{m_2}) \sqcup \\ & \cdots \sqcup (\mathsf{PONEGP}_{1_n} \sqcap \mathsf{PONEGP}_{2_n} \sqcap \cdots \sqcap \mathsf{PONEGP}_{m_n}) \end{aligned}$$

Let us choose a generic PONEGP_{j_i}, for each disjunct we can then refine $\top$ by means of ρ (*Add conjunct* case in Def. 4.10):

$$\bigsqcup_{i=1}^{n} \mathsf{PONEGP}_{j_i} \in \rho_{\sqcap}(\top)$$

then, by proceeding left to right, each disjunct can be specialized (using $\rho_{\sqcup}$ - Specialize disjunct case) by means of $\rho_{\sqcap}$ (again using Add conjunct), until C is obtained.

Now supposing that the thesis holds for $\mathsf{depth}(C) = n - 1$ and let us prove it for $\mathsf{depth}(C) = n$.

From $\top$ let us proceed like in base induction step, with some slight differences. By means of $\rho_{\sqcap}$, let us generate a disjunction of n concepts arbitrarily chosen as above among the atomic concepts (and negated atomic ones) making up every C_i. We obtain again:

$$\bigsqcup_{i=1}^{n} \mathsf{PONEGP}_{j_i} \in \rho_{\sqcap}(\top)$$

Now, by combining $\rho_{\sqcup}$ with $\rho_{\sqcap}$, we can specialize each disjunct (one disjunct per time) adding all the needed atomic concepts and atomic negations as they appeared in every C_i. Now (again chaining $\rho_{\sqcup}$ and $\rho_{\sqcap}$) we start adding the concept $\exists R.\top$ to each C_i, one for each role R in a top-level existential restriction occurring in C_i. Note that all the existential restrictions in C_i have a depth of at most n, therefore all their filler concepts have a depth of at most $n - 1$, thus they belong to $\rho^*(\top)$. Recalling that we chose just one existential restriction per role in each disjunct C_i, we should repeat this procedure for every disjunct and for each existential restriction on the same role therein. This again amounts to chaining $\rho_\sqcup$ with $\rho_\sqcap$.

The remaining value restrictions are unique for each disjunct and role R within it. Unfortunately one cannot simply add $\forall R.\top$ and proceed like for the existential restrictions since $\top \notin \mathcal{DL}-\mathcal{ATOMS}$. So let us indicate with $\forall R.D_{R_i}$ the value restrictions on role R, for each i. We already know that D_{R_i} is in $\mathcal{ALC}$ normal form; let us (with a slight patience effort) proceed by induction on its depth. We have to prove that every concept E in every restriction $\forall R.E$ can be written as a combination of elements of $\mathcal{DL}-\mathcal{ATOMS}$ or obtained by downward refining a value restriction $\forall R.E'$ with $E' \in \mathcal{DL}-\mathcal{ATOMS}$.

$\mathsf{depth}(D_{R_i}) = 0$

Every D_{R_i} of this depth can be written as a conjunction or disjunction of elements in $\mathcal{DL}-\mathcal{ATOMS}$, indeed it is what was showed above. So the proposition is true for $\mathsf{depth}(C) = n$ with value restriction depth at most one.

Now suppose that this holds for $\mathsf{depth}(D_{R_i}) = m-1$ and let us prove it for $\mathsf{depth}(D_{R_i}) = m$ (recall that $m \leq n$ from the outer induction).

Let our E' be what we obtain extracting from D_{R_i} an atomic concept (or a negated one) from each disjunct and putting all them in a disjunction.[21] Now let us chain $\rho_\sqcup$ and $\rho_\sqcap$ refining each disjunct by adding, as usual, one $\exists R.\top$ for each role and existential restriction in it. Such restrictions in D_{R_i} have a depth of at most $m-1$, which is less than n, therefore, by refining each $\exists R.\top$, we obtain the same restrictions occurring in D_{R_i}. There still remain the inner value restrictions, yet since they have at most $m-1$ depth, by the hypotheses of the inner induction, they can be obtained by downward refining a value restriction $\forall R.E'$, $E' \in \mathcal{DL}-\mathcal{ATOMS}$. □

Proposition 4.5. *ρ in Def. 4.10 is complete.*

Proof: We should show that, given C a concept in $\mathcal{ALC}$, $\forall E, E \sqsubset C : E \in \rho^*(C)$.[22]

Suppose, without loss of generality, that C is in $\mathcal{ALC}$-normal form. From Def. 3.1, we know that $E \sqsubset C \Leftrightarrow E \equiv C \sqcap E$. Let D be the equivalent concept w.r.t. E in $\mathcal{ALC}$-normal form. Then we can write:

$$E \equiv \bigsqcup_{i=1}^{n} C_i \sqcap \bigsqcup_{j=1}^{m} D_j$$

Using distributive property of $\sqcap$ with respect to $\sqcup$ we have:

$$E \equiv \bigsqcup_{i=1}^{n} \left(C_i \sqcap \bigsqcap_{j=1}^{m} D_j \right)$$

Let us examine the generic $C_i \sqcap \bigsqcap_{j=1}^{m} D_j$. This can easily be reached from C_i by means of $\rho_\sqcap$ by adding, first of all, all the atomic concepts and negations present in all D_j's; then, for each value restriction, one can add them directly as shown in Proposition 4.4. Successively, we can add $\exists R.\top$ once per role and then specialize until we get the desired filler concept. This is possible, again as shown in Proposition 4.4, as ρ is weakly complete, hence every concept can reached starting from $\top$ by downward refinement. Exploiting the definition of $\rho_\sqcup$, we can refine a concept in $\mathcal{ALC}$-normal form in a *disjunct-wise* fashion (one disjunct at a time); therefore, combining $\rho_\sqcup$ with $\rho_\sqcap$, D can be reached starting from C by means of a refinement chain. Hence ρ is complete. □

Proposition 4.6. *δ in Def. 4.11 is weakly complete.*

Proof: (Sketch) The generic C can be supposed in $\mathcal{ALC}$-normal form without loss of generality. Therefore, let us write $C = \bigsqcup_{i=0}^{n} C_i$. In order to prove this proposition, one can follow the induction schema employed for the dual operator ρ (see Proposition 4.4), with the appropriate changes. First, we should start from $\bot$ instead of $\top$. Then we recall that the roles of $\exists$ and $\forall$ may be perfectly inverted. Furthermore, for proving that any $\exists R.C$ can be obtained by means of δ, recall that any $\exists R.E$ can be obtained directly from $\bot$ by applying once $\delta_\sqcup$ up to the minimum depths at which value restrictions occur in E, then adding $\forall R.\bot$, and finally refining each universal restriction. □

Proposition 4.7. *δ in Def. 4.11 is complete.*

Proof: (Informal) Following the schema in Proposition 4.5, we just use the relation below that holds if $C \sqsubset D$

$$D \equiv C \sqcup D'$$

where D' stands for D in $\mathcal{ALC}$-normal form. Notice that $C \sqcup D'$ is again in normal form. Using the results of Proposition 4.6 and the definition of δ one can derive the completeness of δ inductively on the length of D' □

Proposition 5.1. *Consider a learning problem where $E = E^+ \cup E^-$ is the set of examples and counterexamples of the concept C to be learnt. Let $E_C = E_C^+ \cup E_C^-$ be the set obtained by calculating concept level representatives of*

[21] This is exactly the same that we have done twice in the outer induction above.

[22] Or, better, $\forall E, E \sqsubset C : \exists E' \in \rho^*(C) \wedge E \equiv E'$.

the corresponding instances, respectively, in E^+ and E^-[23]*, with $E_C^+ \cap E_C^- = \emptyset$. Let $\mathcal{H}_S$ be the (possibly infinite) set of hypotheses solving the learning problem. Then, it holds that:*

$$\mathsf{lcs}(E_i^+) \in \mathcal{H}_S \vee \mathcal{H}_S = \emptyset$$

where E_i^+ are all the elements of E^+ and the lcs *operator returns the least concept subsuming its arguments* [2].

Proof: Recall that lcs of a set of concepts is the minimal concept, with respect to subsumption, that subsumes its arguments. In this case, it is the least possible generalization of the representatives of positive examples. Suppose it is not a solution, that would mean that there is some negative example representative that is subsumed by the lcs. If it also holds that $\mathcal{H}_S \neq \emptyset$ this would mean that $\exists H \in \mathcal{H}_S$ s.t. $\forall E_i^+ \in E^+ \wedge \forall E_i^- \in E^-: \; H \sqsubseteq E_i^+ \wedge H \not\sqsubseteq E_i^-$. From the definition of lcs, we have that $H \sqsupseteq \mathsf{lcs}(E_i^+)$ and, being $\mathsf{lcs} \notin \mathcal{H}_S$, we have that $\exists E_j^- \in E^-: \; \mathsf{lcs}(E_i^+) \sqsupseteq E_j^-$. However, by the transitive property of subsumption, we have also that $H \sqsupseteq E_j^-$, hence $H \notin \mathcal{H}_S$. Therefore we have proved that if $\mathsf{lcs}(E_i^+) \notin \mathcal{H}_S$ then $\mathcal{H}_S = \emptyset$ that is what we claimed. □

References

1. Irina Astrova (2004) Reverse engineering of relational databases to ontologies. In: Christoph Bussler, John Davies, Dieter Fensel, Rudi Studer (eds), ESWS, vol 3053 of lecture notes in computer science, Springer, pp 327–341
2. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider P (eds) (2003) The description logic handbook. Cambridge University Press
3. Baader F, Küsters R (2005) Non-standard inferences in description logics: The story so far. In: Gabbay D, Goncharov SS, Zakharyaschev M (eds), Mathematical problems from applied logic. New logics for the XXIst century, vol 4 of International mathematical series. Kluwer/Plenum Publishers (To appear)
4. Badea L, Nienhuys-Cheng S-H (2000) A refinement operator for description logics. In: Cussens J, Frisch A (eds) Proceedings of the 10th international conference on inductive logic programming, vol 1866 of LNAI, pp 40–59
5. Liviu Badea, Monica Stanciu (1999) Refinement operators can be (weakly) perfect. In: Saso Dzeroski, Peter AF (eds) ILP, vol 1634 of lecture notes in computer science, Springer, pp 21–32
6. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Scientific American
7. Alex Borgida, Thomas JW, Haym Hirsh (2005) Towards measuring similarity in description logics. In: Ian Horrocks, Ulrike Sattler, Frank Wolter (eds) Description logics 2005. Proceedings of the 2005 international workshop on description logics (DL-2005), Edinburgh, UK, vol 147. CEUR-WS
8. Cohen WW, Hirsh H (1994) The learnability of description logics with equality constraints. Mach Learn 17(2–3):169–199
9. Cohen WW, Hirsh H (1994) Learning the classic description logic: Theoretical and experimental results. In: Torasso P, Doyle J, Sandewall E (eds) Proceedings of the 4th international conference on the principles of knowledge representation and reasoning, Morgan Kaufmannpp, pp 121–133
10. d'Amato C, Fanizzi N, Esposito F (2005) A semantic similarity measure for expressive description logics. In: Pettorossi A (ed) Proceedings of convegno Italiano di logica computazionale (CILC05), Rome, Italy, http://www.disp.uniroma2.it/CILC2005/downloads/papers/15.dAmato_CILC05.%pdf.
11. d'Amato C, Fanizzi N, Esposito F (2006) A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In: Proceedings of the 21st annual ACM symposium of applied computing, SAC2006, Dijon, France
12. Dietterich TG, London RL, Clarkson K, Dromey G (1982) The Handbook of artificial intelligence, vol III, Chapter XIV— Learning and inductive inference, William Kaufmann, Los Altos, CA, pp 323–512
13. Donini FM, Lenzerini M, Nardi D, Nutt W (1998) An epistemic operator for description logics. Artif Intell 100(1–2):225–274
14. Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, Semeraro G (2004) Induction and revision of terminologies. In: Lorenza Saitta (ed) Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), IOS Press, pp 1007–1008
15. Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, Giovanni Semeraro (2005) A counterfactual-based learning algorithm for description logic. In: Proceedings of the Italian conference on artificial intelligence, AI*IA2005, pp 406–417
16. Floriana Esposito, Donato Malerba, Francesca A. Lisi (2000) Machine learning for intelligent processing of printed documents. J Intell Inf Syst 14(2–3):175–198
17. Stefano Ferilli, Nicola Di Mauro, Teresa Maria Altomare Basile, Floriana Esposito (2003) Incremental induction of rules for document image understanding. In: Cappelli A, Turini F (eds) Proceedings of the conference of the Italian association for artificial intelligence, AI*IA, vol 2829 of Lecture notes in computer science, pp 176–188
18. Gerhard Friedrich, Kostyantyn Shchekotykhin (2005) A general diagnosis method for ontologies. In: Gil Y, Motta V, Benjamins E, Mark A Musen (eds) Proceedings of the 4th international semantic web conference, ISWC2005, no 3279 in LNCS, Galway, Ireland, pp 232–246
19. Thomas R Gruber (1993) A Translation Approach to Portable Ontology Specifications. Academic Press
20. Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, York Sure (2005) A framework for handling inconsistency in changing ontologies. In: Gil Y, Motta V, Benjamins E, Mark A Musen (eds) Proceedings of the 4th international semantic web conference, ISWC2005, no 3279 in LNCS, Galway, Ireland, pp 353–367
21. Ian Horrocks, Peter F Patel-Schneider, Frank van Harmelen (2003) From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J Web Semantics 1(1):7–26
22. Luigi Iannone, Ignazio Palmisano (2005) An algorithm based on counterfactuals for concept learning in the semantic web. In: Moonis Ali, Floriana Esposito (eds) Innovations in applied artificial intelligence, 18th international conference on industrial and engineering applications of artificial intelligence and expert systems, IEA/AIE 2005, vol 3533 of Lecture notes in computer science, pp 370–379
23. Kietz J-U (2002) Learnability of description logic programs. In: Matwin S, Sammut C (eds) Proceedings of the 12th international

[23] Depending on the chosen strategy it could happen that $|E_C^*| < |E^*|$, $E^* \in \{E^+, E^-\}$, i.e. there is a representative for more than one instance.

conference on inductive logic programming, vol 2583 of LNAI, Sydney, pp 117–132
24. Alexander Maedche, Steffen Staab (2001) Ontology learning for the semantic web. IEEE Intell Syst 16(2):March/April
25. Michalski RS, Carbonell JG, Mitchell TM (eds) (1984) Machine learning: An artificial intelligence approach, Chapter a theory and methodology of inductive learning, Berlin, Heidelberg pp 83–130
26. Tom Mitchell (1982) Generalization as search. Artif Intell, Pages 203–226
27. RDF-Schema (2003) RDF Vocabulary Description Language 1.0: RDF Schema. Available at: http://www.w3c.org/TR/rdf-schema
28. Rouveirol C, Ventos V (2000) Towards learning in CARIN-$\mathcal{ALN}$. In: Cussens J, Frisch A (eds) Proceedings of the 10th international conference on inductive logic programming, vol 1866 of LNAI, pp 191–208
29. Schmidt-Schauß M, Smolka G (1991) Attributive concept descriptions with complements. Artif Intell 48(1):1–26
30. Teege G (1994) A subtraction operation for description logics. In: Torasso P, Doyle J, Sandewall E (eds) Proceedings of the 4th international conference on principles of knowledge representation and reasoning, Morgan Kaufmann, pp 540–550
31. Leslie G Valiant (1984) A theory of the learnable. Commun ACM, 27(11):1134–1142
32. Patrick RJ van der Laag, Shan-Hwei Nienhuys-Cheng (1994) Existence and nonexistence of complete refinement operators. In: Francesco Bergadano, Luc De Raedt (eds) ECML, vol 784 of Lecture notes in computer science, pp 307–322
33. Vere SA (1980) Multilevel counterfactuals for generalizations of relational concepts and productions. Artif Intell 14:139–164