Foundations of Refinement Operators for Description Logics

Jens Lehmann[1⋆] and Pascal Hitzler[2⋆⋆]

[1] Universität Leipzig, Department of Computer Science
Johannisgasse 26, D-04103 Leipzig, Germany,
lehmann@informatik.uni-leipzig.de
[2] Universität Karlsruhe (TH), AIFB Institute
D-76128 Karlsruhe, Germany,
hitzler@aifb.uni-karlsruhe.de

Abstract In order to leverage techniques from Inductive Logic Programming for the learning in description logics (DLs), which are the foundation of ontology languages in the Semantic Web, it is important to acquire a thorough understanding of the theoretical potential and limitations of using refinement operators within the description logic paradigm. In this paper, we present a comprehensive study which analyses desirable properties such operators should have. In particular, we show that ideal refinement operators in general do not exist, which is indicative of the hardness inherent in learning in DLs. We also show which combinations of desirable properties are theoretically possible, thus providing an important step towards the definition of practically applicable operators.

1 Introduction

With the advent of the Semantic Web and Semantic Technologies, ontologies are becoming one of the most prominent paradigms for knowledge representation and reasoning. However, recent progress in the field faces a lack of available ontologies due to the fact that engineering such ontologies constitutes a considerable investment of resources. Methods for the automated acquisition of ontologies are therefore required. In this article, we develop theoretical foundations for the creation of such methods.

In 2004, the World Wide Web Consortium (W3C) recommended the Web Ontology Language OWL[3] as a standard for modelling ontologies on the Web. In the meantime, many studies and applications using OWL have been reported in research, many of which go beyond Internet usage and employ the power of

⋆ The first author acknowledges support by the German Federal Ministry of Education and Research (BMBF) under the SoftWiki project (grant 01 ISF02 B).
⋆⋆ The second author acknowledges support by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (grant 01 IMD01 B) and by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.
[3] http://www.w3.org/2004/OWL/

ontological modelling in other fields like software engineering, knowledge management, and cognitive systems.

In essence, OWL coincides with the description logic $\mathcal{SHOIN}(\mathrm{D})$ and is thus a knowledge representation formalism based on first-order logic. In order to leverage machine-learning approaches for the acquisition of OWL ontologies, it is therefore required to develop methods and tools for the learning in description logics. To date, only few investigations have been carried out on this topic, which can be attributed to the fact that description logics (DLs) have only recently become a major paradigm in knowledge representation and reasoning.

In this paper, we investigate the applicability of methods from Inductive Logic Programming (ILP) for the learning in description logic knowledge bases. We are motivated by the success of ILP methods and believe that similar results can be achieved for DLs.

Central to the usual ILP approach are the so-called *refinement operators* which are used to traverse the search space, and many approaches indeed hinge on the definition of a suitable such operator. Theoretical investigations on ILP refinement operators have identified desirable properties for them to have, which impact on their performance. These properties thus provide guidelines for the definition of suitable operators. It turns out, however, that for hard learning settings there are theoretical limitations on the properties a refinement operator can have. A corresponding general analysis therefore provides a clear understanding of the difficulties inherent in a learning setting, and also allows to derive directions for researching suitable operators.

In this paper we therefore give a full analysis of properties of refinement operators for description logics. To the best of our knowledge, such a complete analysis has not been done before, but the need for this investigation was expressed in [6,7]. The main contribution of this article is to derive a fundamental theorem about properties of refinement operators in DLs. This can serve as the foundation for the design of concrete refinement operators, which are used for induction – with potential applications in other areas of Machine Learning like clustering and data mining.

The paper is structured as follows. In Section 2 we will give a brief introduction to description logics. Section 3 formally describes the learning problem in description logics. Refinement operators and their properties are introduced. The main section is Section 4, which contains the results we obtained. We will fully analyse all combinations of interesting properties of refinement operators. This means we will show which combinations of properties are possible, i.e. for which combinations a refinement operator with these properties exists. We make only basic assumptions with respect to the description logic we are looking at, to cover as many description logics as possible. In Section 5 we discuss related work, in particular the relation to refinement operators in the area of traditional Inductive Logic Programming. Finally, in Section 6 we summarise our work and draw conclusions.

Some proofs are omitted for lack of space. They can be found in a separate technical report [12].

2 Description Logics

Description logics represent knowledge in terms of *objects*, *concepts*, and *roles*. Objects correspond to constants, concepts to unary predicates, and roles to binary predicates in first order logic. In description logic systems information is stored in a *knowledge base*, which is a set of axioms. It is divided in (at least) two parts: *TBox* and *ABox*. The ABox contains *assertions* about objects. It relates objects to concepts and roles. The TBox describes the *terminology* by relating concepts and roles.

We briefly introduce the $\mathcal{ALC}$ description logic, which is the target language of our learning algorithm and refer to [1] for further background on description logics. Syntax and semantic of $\mathcal{ALC}$ concept constructors is shown in Table 1. As usual in logics, interpretations are used to assign a meaning to syntactic constructs. Let N_I denote the set of objects, N_C denote the set of atomic concepts, and N_R denote the set of roles. An *interpretation* $\mathcal{I}$ consists of a non-empty *interpretation domain* $\Delta^\mathcal{I}$ and an *interpretation function* $\cdot^\mathcal{I}$, which assigns to each object $a \in N_I$ an element of $\Delta^\mathcal{I}$, to each concept $A \in N_C$ a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, and to each role $r \in N_R$ a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. Interpretations are extended to concepts as shown in Table 1, and to other elements of a knowledge base in a straightforward way. An interpretation, which satisfies an axiom (set of axioms) is called a model of this axiom (set of axioms). An $\mathcal{ALC}$ concept is in *negation normal form* if negation only occurs in front of concept names. $\mathcal{SHOIN}(D)$, the description language corresponding to OWL (to be precise it corresponds to the dialect OWL DL), is an extension of $\mathcal{ALC}$. The results presented in this paper are general, i.e. they hold for all reasonable expressive description logics. (The criteria these languages have to fulfil are described later in Definition 3.)

construct	syntax	semantics
atomic concept	A	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
role	r	$r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$
negation	$\neg C$	$(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$
existential	$\exists r.C$	$(\exists r.C)^\mathcal{I} = \{a \mid$ $\exists b.(a,b) \in r^\mathcal{I}$ and $b \in C^\mathcal{I}\}$
universal	$\forall r.C$	$(\forall r.C)^\mathcal{I} = \{a \mid$ $\forall b.(a,b) \in r^\mathcal{I}$ implies $b \in C^\mathcal{I}\}$

Table 1. $\mathcal{ALC}$ syntax and semantics

It is the aim of *inference algorithms* to extract implicit knowledge from a given knowledge base. Standard reasoning tasks include *instance checks*, *retrieval* and *subsumption*. We will only explicitly define the latter. Let C, D be concepts

and $\mathcal{T}$ a TBox. *C is subsumed by D*, denoted by $C \sqsubseteq D$, iff for any interpretation $\mathcal{I}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. *$C$ is subsumed by D with respect to $\mathcal{T}$* (denoted by $C \sqsubseteq_{\mathcal{T}} D$) iff for any model $\mathcal{I}$ of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. *$C$ is equivalent to D (with respect to $\mathcal{T}$)*, denoted by $C \equiv D$ ($C \equiv_{\mathcal{T}} D$), iff $C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) and $D \sqsubseteq C$ ($D \sqsubseteq_{\mathcal{T}} C$). *$C$ is strictly subsumed by D (with respect to $\mathcal{T}$)*, denoted by $C \sqsubset D$ ($C \sqsubset_{\mathcal{T}} D$), iff $C \sqsubseteq D$ ($C \sqsubseteq_{\mathcal{T}} D$) and not $C \equiv D$ ($C \equiv_{\mathcal{T}} D$).

3 Learning in Description Logics using Refinement Operators

In this section we will briefly describe the learning problem in description logics. The process of learning in logics, i.e. finding logical explanations for given data, is also called *inductive reasoning*. In a very general setting this means that we have a logical formulation of background knowledge and some observations. We are then looking for ways to extend the background knowledge such that we can explain the observations, i.e. they can be deduced from the modified knowledge. More formally we are given background knowledge B, positive examples E^+, negative examples E^- and want to find a hypothesis H such that from H together with B the positive examples follow and the negative examples do not follow. It is not required that the same logical formalism is used for background knowledge, examples, and hypothesis, but often this is the case.

Definition 1 (learning problem in description logics). *Let a concept name* `Target`*, a knowledge base $\mathcal{K}$ (not containing* `Target`*), and sets E^+ and E^- with elements of the form* `Target`(a) *($a \in N_I$) be given. The learning problem is to find a concept C such that* `Target` *does not occur in C and for $\mathcal{K}' = \mathcal{K} \cup \{\texttt{Target} \equiv C\}$ we have $\mathcal{K}' \models E^+$ and $\mathcal{K}' \not\models E^-$.*

The goal of learning is to find a correct concept with respect to the examples. This can be seen as a search process in the space of concepts. A natural idea is to impose an ordering on this search space and use operators to traverse it. This idea is well-known in Inductive Logic Programming [18], where refinement operators are widely used to find hypotheses. Intuitively, downward (upward) refinement operators construct specialisations (generalisations) of hypotheses.

Definition 2. *A* quasi-ordering *is a reflexive and transitive relation. Let S be a set and $\preceq$ a quasi-ordering on S. In the quasi-ordered space $(S, \preceq)$ a* downward (upward) refinement operator *ρ is a mapping from S to 2^S, such that for any $C \in S$ we have that $C' \in \rho(C)$ implies $C' \preceq C$ ($C \preceq C'$). C' is called a* specialisation *(*generalisation*) of C.*

This idea can be used for searching in the space of concepts. As ordering we can use subsumption. (Note that the subsumption relation $\sqsubseteq$ is a quasi-ordering.) If a concept C subsumes a concept D ($D \sqsubseteq C$), then C will cover all examples, which are covered by D. This makes subsumption a suitable order for searching in concepts. In this section we will analyse refinement operators for concepts

with respect to subsumption and a description language $\mathcal{L}$, which we will call $\mathcal{L}$ refinement operators in the sequel.

Definition 3. *Let $\mathcal{L}$ be a description language, which allows to express $\top$, $\bot$, conjunction, disjunction, universal quantification, and existential quantification. A refinement operator in the quasi-ordered space $(\mathcal{L}, \sqsubseteq)$ is called an* $\mathcal{L}$ refinement operator.

We need to introduce some notions for refinement operators.

Definition 4. *A* refinement chain of an $\mathcal{L}$ refinement operator ρ of length n *from a concept C to a concept D is a finite sequence $C_0, C_1, \ldots, C_n$ of concepts, such that $C = C_0, C_1 \in \rho(C_0), C_2 \in \rho(C_1), \ldots, C_n \in \rho(C_{n-1}), D = C_n$. This refinement chain* goes through *E iff there is an i ($1 \leq i \leq n$) such that $E = C_i$. We say that D can be reached from C by ρ if there exists a refinement chain from C to D. $\rho^*(C)$ denotes the set of all concepts, which can be reached from C by ρ. $\rho^m(C)$ denotes the set of all concepts, which can be reached from C by a refinement chain of ρ of length m.*

Definition 5. *A concept C is a* downward cover *of a concept D iff $C \sqsubset D$ and there does not exist a concept E with $C \sqsubset E \sqsubset D$. A concept C is an* upward cover *of a concept D iff $D \sqsubset C$ and there does not exist a concept E with $D \sqsubset E \sqsubset C$.*

If we look at refinements of an operator ρ we will often write $C \rightsquigarrow_\rho D$ instead of $D \in \rho(C)$. If the used operator is clear from the context it is usually omitted, i.e. we write $C \rightsquigarrow D$.

We will introduce the concept of weak equality of concepts, which is similar to equality of concepts, but takes into account that the order of elements in conjunctions and disjunctions is not important. By equality of two concepts we mean that the concepts are syntactically equal. Equivalence of two concepts means that the concepts are logically equivalent (see preliminaries). Weak equality of concepts is coarser than equality and finer than equivalence (viewing the equivalence, equality, and weak equality of concepts as equivalence classes).

Definition 6. *We say that the concepts C and D are* weakly (syntactically) equal*, denoted by $C \simeq D$ iff they are equal up to permutation of arguments of conjunction and disjunction. Two sets S_1 and S_2 of concepts are weakly equal if for any $C_1 \in S_1$ there is a $C_1' \in S_2$ such that $C_1 \simeq C_1'$ and vice versa.*

Refinement operators can have certain properties, which can be used to evaluate their usefulness for learning hypotheses. These properties are what we investigate in this paper.

Definition 7. *An $\mathcal{L}$ refinement operator ρ is called*

- (locally) finite *iff $\rho(C)$ is finite for any concept C.*

- (syntactically) redundant *iff there exists a refinement chain from a concept C to a concept D, which does not go through some concept E and a refinement chain from C to a concept weakly equal to D, which does go through E.*
- proper *iff for all concepts C and D, $D \in \rho(C)$ implies $C \not\equiv D$.*
- ideal *iff it is finite, complete (see below), and proper.*

An $\mathcal{L}$ downward refinement operator ρ is called

- complete *iff for all concepts C and D with $C \sqsubset D$ we can reach a concept E with $E \equiv C$ from D by ρ.*
- weakly complete *iff for all concepts C with $C \sqsubset \top$ we can reach a concept E with $E \equiv C$ from $\top$ by ρ.*
- minimal *iff for all C, $\rho(C)$ contains only downward covers and all its elements are incomparable with respect to $\sqsubseteq$.*

The corresponding notions for upward refinement operators are defined dually.

4 Analysing the Properties of Refinement Operators

In this section we will analyse the properties of refinement operators in description logics. In particular, we are interested in seeing which desired properties can be combined in a refinement operator and which properties are impossible to combine. This is interesting for two reasons: The first one is that this gives us a good impression of how hard (or easy) it is to learn concepts. The second reason is that this can also serve as a practical guide for designing refinement operators. Knowing the theoretical limits allows the designer of a refinement operator to focus on achieving the best possible properties.

$\mathcal{ALC}$ refinement operators have been designed in [6,9]. However, a full theoretical analysis for DL refinement operators has not been done to the best of our knowledge (not even for a specific language). Therefore all propositions in this section are new unless explicitly mentioned otherwise.

As a first property we will briefly analyse minimality of $\mathcal{L}$ refinement operators, in particular the existence of upward and downward covers in $\mathcal{ALC}$. It is not immediately obvious that e.g. downward covers exist in $\mathcal{ALC}$, because it could be the case that for any concept C and D with $C \sqsubset D$ one can always construct a concept E with $C \sqsubset E \sqsubset D$. However, it is possible to show that downward covers do exist.

Proposition 1. *Downward (upward) covers of $\top$ ($\bot$) exist in $\mathcal{ALC}$.*

Example 1. The following is a downward cover of the $\top$ concept:

$$\bigsqcup_{r \in N_R} \exists r.\top \sqcup \bigsqcup_{A \in N_C} A$$

This means that non-trivial minimal operators, i.e. operators which do not map every concept to the empty set, can be constructed. However, minimality

of refinement steps is not a directly desired goal in general. Minimal operators are in some languages more likely to lead to overfitting, because they may not produce sufficient generalisation leaps. This is a problem in languages which are closed under boolean operations, i.e. $\mathcal{ALC}$ and more expressive languages.

Indeed, the following result suggests that minimality may not play a central role for DL refinement operators, as it is incompatible with (weak) completeness. A weaker result was claimed to hold, but not proved, in [4]. We formulate our result for the description logic $\mathcal{AL}$, the concepts of which are inductively defined as follows: $\top$, $\bot$, $\exists r.\top$, A, $\neg A$ with $A \in N_C$, $r \in N_R$ are $\mathcal{AL}$ concepts. If C and D are $\mathcal{AL}$ concepts, then $C \sqcap D$ is an $\mathcal{AL}$ concept. If C is an $\mathcal{AL}$ concept and r a role, then $\forall r.C$ is an $\mathcal{AL}$ concept.

Proposition 2. *There exists no minimal and weakly complete $\mathcal{AL}$ downward refinement operator.*

In the sequel we analyse desired properties of $\mathcal{L}$ refinement operators: completeness, properness, finiteness, and non-redundancy. We show several positive and negative results, which together yield a full analysis of these properties.

Proposition 3. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a complete and finite $\mathcal{L}$ refinement operator.*[4]

In particular, the following operator can be shown to be complete and finite for any language $\mathcal{L}$ we consider:

$$\rho(C) = \{C \sqcap \top\} \cup \{D \mid |D| \leq (\text{number of } \top \text{ occurrences in } C) \text{ and } D \sqsubset C\}$$

Of course, it is obvious that the operator used to prove Proposition 3 is not useful in practise, since it merely generates concepts without paying attention to efficiency. However, in [11], we have developed a complete and finite operator, which was integrated into the Genetic Programming framework and shown to be useful in a preliminary evaluation.

Let it be noted that in many scenarios it appears to be quite difficult to design a good complete and finite refinement operator. The reason is that finiteness can only be achieved by using non-proper refinement steps. We will now show that it is impossible to define a complete, finite, and proper refinement operator. Such operators are known as ideal and their non-existence indicates that learning concepts in sufficiently expressive description logics is hard.

Proposition 4. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there does not exist any ideal $\mathcal{L}$ refinement operator.*

We will give a proof sketch: By contradiction, we assume that there exists an ideal downward refinement operator ρ. We further assume that there is a role

[4] Our result in fact invalidates a claim made in [4] stating that there can be no complete and finite $\mathcal{ALER}$ refinement operator.

$r \in N_R$. Let $\rho(\top) = \{C_1, \ldots, C_n\}$ be a set of refinements of the $\top$ concept. (This set has to be finite, since ρ is finite.) Let m be a natural number larger than the maximum of the *quantifier depths* (depth of the nesting of quantifications) of the concepts in $\rho(\top)$. We construct a concept D as follows:

$$D = \underbrace{\forall r. \ldots . \forall r}_{m-\text{times}} . \bot \sqcup \underbrace{\exists r. \ldots . \exists r}_{(m+1)-\text{times}} . \top$$

We have shown, which is the main part of the proof, that there exists no concept with a quantifier depth smaller than m, which strictly subsumes D and is not equivalent to $\top$. This means that $C_1, \ldots, C_n$ do not subsume D (note that the properness of ρ implies that $C_1, \ldots, C_n$ are not equivalent to $\top$), so D cannot be reached by further refinements from any of these concepts. Since $C_1, \ldots, C_n$ are the only refinements of $\top$, it is impossible to reach D from $\top$. Thus ρ is not complete, which is what we wanted to show.

However, if the requirement of finiteness is dropped, corresponding operators exist.

Proposition 5. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a complete and proper $\mathcal{L}$ refinement operator.*

Propositions 3, 4, and 5 state that for complete refinement operators, which are usually desirable, one has to sacrifice properness or finiteness. Both combinations can be useful in practise. As noted above, we have developed a complete and finite operator in [11], because the finiteness property was important in this context. However, in [13] we have chosen to develop a complete and proper operator, in an ILP style top-down learning algorithm, because it is easier to overcome the problem of an infinite operator in this context.

We will now look at non-redundancy.

Proposition 6. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a complete and non-redundant $\mathcal{L}$ refinement operator.*

Proof. We will prove the result by showing that each complete operator can be transformed to a complete and non-redundant operator. Note that in the following, we will use the role r to create concepts with a certain depth. If N_R does not contain any role, the desired effect can also be achieved by using conjunctions or disjunctions of $\top$ and $\bot$, but this would render the proof less readable.

We will use the fact that the set of concepts in $\mathcal{L}$ is countably infinite. The countability already follows from the fact that there is just a finite number of concepts with a given length. Hence, we can divide the set of all concepts in finite subsets, where each subset contains all concepts of the same length. We can then start enumerating concepts starting with the subset of concepts of length 1, then length 2 etc. Thus, there is a bijective function $f : \mathcal{L} \mapsto \mathbf{N}$, which assigns a different number to each concept $C \in \mathcal{L}$. We denote the inverse function mapping numbers to concepts by f^{inv}.

Now, we modify a given complete refinement operator ρ, e.g. the operator in the proof of Proposition 5 (see [12] for details), in the following way: For any concept C, $\rho(C)$ is modified by changing any element $D \in \rho(C)$ with depth d to

$$D \sqcap \underbrace{\forall r. \ldots \forall r}_{d+1 \text{ times}} . (\underbrace{\top \sqcap \cdots \sqcap \top}_{f(C) \text{ times}})$$

We claim that the resulting operator, which we want to denote by ρ' is complete and non-redundant.

The completeness of ρ' follows from the completeness of ρ, since the construct we have added does not change the meaning (it is equivalent to $\top$).

To prove non-redundancy, we will first define a function ρ^{inv}, which maps conjunctions, which contain at least one element of the form $\forall r. \ldots \forall r.(\top \sqcap \cdots \sqcap \top)$, to concepts:

$$\underbrace{\rho^{\text{inv}}(C \sqcap \forall r. \ldots \forall r. \underbrace{(\top \sqcap \cdots \sqcap \top)}_{n \text{ times}}}_{\text{element with largest depth}} = f^{\text{inv}}(n)$$

We can see that $D \in \rho'(C)$ implies $\rho^{\text{inv}}(D) = C$, so ρ^{inv} allows to invert a refinement step of ρ'. Furthermore, we have $C \simeq D$ implies $\rho^{\text{inv}}(C) = \rho^{\text{inv}}(D)$, because ρ^{inv} treats all weakly equal concepts in the same way.

By the definition of redundancy, there needs to be a concept C and concepts D_1, D_2 with $D_1 \simeq D_2$, such that there is a refinement chain from C to D_1 and a different refinement chain from C to D_2. However, if D_1 and D_2 are weakly equal, then $\rho^{\text{inv}}(D_1) = \rho^{\text{inv}}(D_1)$, and by continuing to apply ρ^{inv} we will reach C. Hence the two refinement chains from C to D_1 and D_2, respectively, cannot be different. Thus, ρ' is non-redundant.

Essentially, the proof of Proposition 6 is done by showing how to make complete operators non-redundant by using the fact that the set of concepts in any considered language $\mathcal{L}$ is countably infinite. We note, however, that under a mild additional assumption Proposition 6 no longer holds.

Proposition 7. *Let ρ be an arbitrary $\mathcal{L}$ refinement operator, where $\mathcal{L}$ satisfies the conditions stated in Definition 3, and for any concept C, $\rho^*(C)$ contains only finitely many different concepts equivalent to $\bot$. Then ρ is not complete and non-redundant.*

The assumption made in the proposition is indeed mild: If we have $\bot \in \rho^*(C)$ for any concept C, then it is already satisfied. The assumption is made in order to disallow pure theoretical constructions, as in the proof of Proposition 6, which use syntactic concept extensions, that do not alter the semantics of a concept, to ensure non-redundancy. In fact, the result also holds under an even milder, but more technical assumption, see our report [12].

As a consequence, completeness and non-redundancy usually cannot be combined. It is often desirable to have (weakly) complete operators, but in order to

have a full analysis of $\mathcal{L}$ refinement operators we will now also investigate incomplete operators.

Proposition 8. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a finite, proper, and non-redundant $\mathcal{L}$ refinement operator.*

Proof. The following operator has the desired properties:

$$\rho(C) = \begin{cases} \{\bot\} & \text{if } C \not\equiv \bot \\ \emptyset & \text{otherwise} \end{cases}$$

It is obviously finite, because it maps concepts to sets of cardinality at most 1. It is non-redundant, because it only reaches the bottom concept and there exists no refinement chain of length greater than 2. It is proper, because all concepts, which are not equivalent to the bottom concept strictly subsume the bottom concept.

The corresponding upward operator is:

$$\phi(C) = \begin{cases} \{\top\} & \text{if } C \not\equiv \top \\ \emptyset & \text{otherwise} \end{cases}$$

The arguments for its finiteness, properness, and non-redundancy are analogous to the downward case.

We can now summarise the results we have obtained so far.

Theorem 1. *Let $\mathcal{L}$ be a language, which satisfies the conditions stated in Definition 3. Considering the properties completeness, properness, finiteness, and non-redundancy the following are maximal sets of properties (in the sense that no other of the mentioned properties can be added) of $\mathcal{L}$ refinement operators:*

1. $\{complete, finite\}$
2. $\{complete, proper\}$
3. $\{non\text{-}redundant, finite, proper\}$

All results hold under the mild hypothesis stated in Proposition 7.

A property we have not yet considered is weak completeness. Usually weak completeness is sufficient, because it allows to search for a good concept starting from $\top$ downwards (top-down approach) or from $\bot$ upwards (bottom-up approach).

We will see that we get different results when considering weak completeness instead of completeness.

Proposition 9. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a weakly complete, non-redundant, and proper $\mathcal{L}$ refinement operator.*

The result just given also holds when properness is replaced by finiteness.

Proposition 10. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists a weakly complete, non-redundant, and finite $\mathcal{L}$ refinement operator.*

However, properness and finiteness cannot be achieved at the same time if weak completeness is imposed. To show this, we can use the same proof as for Proposition 8, where we have shown that for any finite and proper refinement operator, there exists a concept, which cannot be reached from $\top$.

Corollary 1. *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 3, there exists no weakly complete, finite, and proper $\mathcal{L}$ refinement operator.*

The result of the previous observations is that, when requiring only weak completeness instead of completeness, non-redundant operators are possible.[5]

The following theorem is the result of the full analysis of the desired properties of $\mathcal{L}$ refinement operators.

Theorem 2 (Property Theorem). *Let $\mathcal{L}$ be a language, which satisfies the conditions stated in Definition 3. Considering the properties completeness, weak completeness, properness, finiteness, and non-redundancy the following are maximal sets of properties (in the sense that no other of the mentioned properties can be added) of $\mathcal{L}$ refinement operators:*

1. $\{$*weakly complete*, *complete*, *finite*$\}$
2. $\{$*weakly complete*, *complete*, *proper*$\}$
3. $\{$*weakly complete*, *non-redundant*, *finite*$\}$
4. $\{$*weakly complete*, *non-redundant*, *proper*$\}$
5. $\{$*non-redundant*, *finite*, *proper*$\}$

All results hold under the mild hypothesis stated in Proposition 7.

Remark 1. Instead of using subsumption ($\sqsubseteq$) as an ordering over concepts, we can also use subsumption with respect to a TBox $\mathcal{T}$ ($\sqsubseteq_{\mathcal{T}}$). Theorem 2 will also hold in this case.

The results also hold if we consider equality ($=$) instead of weak equality ($\simeq$) as equivalence relation on concepts.

[5] Reducing completeness to weak completeness to obtain non-redundancy seems to be an interesting option for the development of practical operators. However, this approach is problematic since it is not immediately clear how this idea could be put into practise. For instance, a weakly complete and non-redundant downward refinement operator ρ cannot allow refinements of the form $C \rightsquigarrow C \sqcap \top$ or $C \rightsquigarrow C \sqcap \rho(\top)$: A weakly complete operator, which allows one of these refinement steps is also complete and is therefore usually redundant by Proposition 7.

5 Related Work

Related work can essentially be divided in two parts. The first part is research which is directly connected to learning in description logics. The second part is research about refinement operators in general, often connected with the learning of logic programs. We will describe both in turn.

In [4] a refinement operator for $\mathcal{ALER}$ has been designed to obtain a top-down learning algorithm for this language. Properties of refinement operators in this language were discussed and some claims were made, but a full formal analysis was not performed. Our work generalises the results in this article, refutes them in one case, investigates more property combinations, and proves each claim. In [6,9] learning algorithms for description logics, in particular for the language $\mathcal{ALC}$ were created, which also make use of refinement operators. Instead of using the classical approach of combining refinement operators with a search heuristic, they developed an example driven learning method. [6] stated that an investigation of the properties of refinement operators in description logics, as we have done in this article, is required. In [7] downward refinement for $\mathcal{ALN}$ was analysed using a clausal representation of DL concepts. This article also states that further investigation of the properties of refinement operators in description logics is required. Refinement operators have also been dealt with in hybrid systems. In [14] ideal refinement for learning $\mathcal{AL}$-log, a language that merges DATALOG and $\mathcal{ALC}$, was investigated. Based on the notion of $\mathcal{B}$-subsumption, an ideal refinement operator was created. In [5,10] learning algorithms for description logics without refinement operators were analysed.

In the area of Inductive Logic Programming [18] considerable efforts have been made to analyse the properties of refinement operators. Note, that in general using refinement operators for clauses to learn in description logics is possible, but usually not a good choice as shown in [4]. However, the theoretical foundations of refinement operators also apply to description logics, which is why we want to mention work in this area here.

A milestone in Machine Learning [15] in general was the Model Inference System in [19]. Shapiro describes how refinement operators can be used to adapt a hypothesis to a sequence of examples. Afterwards, refinement operators became widely used as (part of) a learning method. [20] have found some general properties of refinement operators in quasi-ordered spaces. Nonexistence conditions for ideal refinement operators relating to infinite ascending and descending refinement chains and covers have been developed. This has been used earlier to show that ideal refinement operators for clauses ordered by θ-subsumption do not exist [20]. Unfortunately, we could not make use of these results, because proving properties of covers in description logics without the restriction to a specific language is likely to be harder than directly proving the results.

[17] discussed refinement for different versions of subsumption, in particular weakenings of logical implication. In [16] it was shown how to extend refinement operators to learn general prenex conjunctive normal form. Perfect, i.e. weakly complete, locally finite, non-redundant, and minimal operators, were discussed in [2]. Since such operators do not exist for clauses ordered by θ-subsumption

[20], weaker versions of subsumption were considered. This was later extended to theories, i.e. sets of clauses [8]. A less widely used property of refinement operators, called flexibility, was discussed in [3]. Flexibility essentially means that previous refinements of an operator can influence the choice of the next refinement. The article discusses how flexibility interacts with other properties and how it influences the search process in a learning algorithm.

6 Conclusions

We have presented a comprehensive analysis of properties of refinement operators. The results are summarised in Theorems 1 and 2. In particular, we have shown that ideal refinement operators for description logics cannot exist. We have also shown in detail which combinations of properties are in general achievable. This analysis is fundamental for using refinement operators in description logics and was requested in [6,7].

After the derivation of these results, two learning systems have been developed, which use the Property Theorem as theoretical foundation. They are reported on elsewhere [11,13]. Both systems are fully implemented and have shown promising results in evaluations.

References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. L. Badea and M. Stanciu. Refinement operators can be (weakly) perfect. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer-Verlag, 1999.
3. Liviu Badea. Perfect refinement operators can be flexible. In Werner Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 266–270. IOS Press, August 2000.
4. Liviu Badea and Shan-Hwei Nienhuys-Cheng. A refinement operator for description logics. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 40–59. Springer-Verlag, 2000.
5. William W. Cohen and Haym Hirsh. Learning the classic description logic: Theoretical and experimental results. In Pietro Torasso Jon Doyle, Erik Sandewall, editor, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 121–133, Bonn, FRG, May 1994. Morgan Kaufmann.
6. Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro. Knowledge-intensive induction of terminologies from metadata. In *The Semantic Web - ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings*, pages 441–455. Springer, 2004.

7. Nicola Fanizzi, Stefano Ferilli, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro. Downward refinement in the ALN description logic. In *4th International Conference on Hybrid Intelligent Systems (HIS 2004), December 2004, Kitakyushu, Japan*, pages 68–73. IEEE Computer Society, 2004.
8. Nicola Fanizzi, Stefano Ferilli, Nicola Di Mauro, and Teresa Maria Altomare Basile. Spaces of theories with ideal refinement operators. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 527–532. Morgan Kaufmann, 2003.
9. Luigi Iannone and Ignazio Palmisano. An algorithm based on counterfactuals for concept learning in the semantic web. In Moonis Ali and Floriana Esposito, editors, *Innovations in Applied Artificial Intelligence*, pages 370–379, Bari, Italy, June 2005. Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.
10. Jörg-Uwe Kietz and Katharina Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14:193–217, 1994.
11. Jens Lehmann. Hybrid learning of ontology classes. In *Proceedings of the 5th International Conference on Machine Learning and Data Mining, MLDM 2007*, 2007.
12. Jens Lehmann and Pascal Hitzler. Foundations of refinement operators for description logics. Technical report, University of Leipzig, 2007. Downloadable from http://www.jens-lehmann.org.
13. Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP)*, 2007.
14. Francesca A. Lisi and Donato Malerba. Ideal refinement of descriptions in AL-log. In Tamás Horváth, editor, *Inductive Logic Programming: 13th International Conference, ILP 2003, Szeged, Hungary, September 29-October 1, 2003, Proceedings*, volume 2835 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2003.
15. Thomas Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.
16. S.-H. Nienhuys-Cheng, W. Van Laer, J. Ramon, and L. De Raedt. Generalizing refinement operators to learn prenex conjunctive normal forms. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 245–256. Springer-Verlag, 1999.
17. S. H. Nienhuys-Cheng, P. R. J. van der Laag, and L. W. N. van der Torre. Constructing refinement operators by decomposing logical implication. In Pietro Torasso, editor, *Advances in Artificial Intelligence: Proceedings of the 3rd Congress of the Italian Association for Artificial Intelligence (AI∗IA '93)*, volume 728 of *LNAI*, pages 178–189, Torino, Italy, October 1993. Springer.
18. Shan-Hwei Nienhuys-Cheng and Ronald de Wolf, editors. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer, 1997.
19. E. Y. Shapiro. Inductive inference of theories from facts. In J. L. Lassez and G. D. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 199–255. The MIT Press, 1991.
20. P. R. J. van der Laag and S-H. Nienhuys-Cheng. Existence and nonexistence of complete refinement operators. In F. Bergadano and L. De Raedt, editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 307–322. Springer-Verlag, 1994.