Foundations of Refinement Operators for Description Logics (Technical Report)

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, 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), it is important to acquire a thorough understanding of the theoretical potential and limitations of using refinement operators within the Descripton Logic paradigm. In this paper, we thus present a comprehensive study which analyses desirable properties such operators should have. In particular, we show that ideal refinement opertors do in general not exist, which is indicative of the hardness inherent in learning 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 and Motivation

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 being sought for.

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

In essence, OWL coincides with the description logic $\mathcal{SHOIN}$(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

[3] http://www.w3.org/2004/OWL/

logics. To date, only very 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 in practice and believe that similar results can be achieved for DLs.

Central to the ILP approach to machine learning are the so-called *refinement operators* which are used to traverse the search space, and an ILP-based approach indeed hinges 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.

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 and their proofs. 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.

2 Description Logics

Description Logics is the name of a family of knowledge representation (KR) formalisms. They emerged from earlier KR formalisms like semantic networks and frames. Their origin lies in the work of Brachman on structured inheritance networks (Brachman, 1978). Since then, Description Logics have enjoyed increasing popularity. They can essentially be understood as fragments of first order predicate logic. They have less expressive power, but usually decidable inference problems and a user-friendly variable free syntax.

Description Logics represent knowledge in terms of *objects*, *concepts*, and *roles*. Concepts formally describe notions in an application domain, e.g. we could define the concept of being a father as "a man having a child". Objects are

members of entities in the application domain and roles are binary relations between objects. 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*. It is divided in 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. (For some expressive description logics this clear separation does not exist.)

As mentioned before, DLs are a family of KR formalisms. In order to simplify the introduction we will introduce the $\mathcal{ALC}$ Description Logic as a prototypical example, because it contains all syntactic constructs we need in this article.

$\mathcal{ALC}$ stands for *attribute language with complement*. It allows to construct complex concepts from simpler ones using various language constructs. The next definition shows how such concepts can be built.

Definition 1 (syntax of $\mathcal{ALC}$ concepts). *Let N_R be a set of* role names *and N_C be a set of concept names ($N_R \cap N_C = \emptyset$). The elements of the latter set are called* atomic concepts. *The set of $\mathcal{ALC}$ concepts is inductively defined as follows:*

1. *Each atomic concept is a concept.*
2. *If C and D are $\mathcal{ALC}$ concepts and $r \in N_R$ a role, then the following are also $\mathcal{ALC}$ concepts:*
   - *$\top$ (top), $\bot$ (bottom)*
   - *$C \sqcup D$ (disjunction), $C \sqcap D$ (conjunction), $\neg C$ (negation)*
   - *$\forall r.C$ (value/universal restriction), $\exists r.C$ (existential restriction)*

The semantics of $\mathcal{ALC}$ concept is defined by interpretations. See the following definition and Table 1.

Definition 2 (interpretation). *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 $A \in N_C$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each $r \in N_R$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.*

Table 1. $\mathcal{ALC}$ semantics

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 concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
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	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
exists restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{a \mid \exists b.(a,b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$(\forall r.C)^{\mathcal{I}} = \{a \mid \forall b.(a,b) \in r^{\mathcal{I}} \text{ implies } b \in C^{\mathcal{I}}\}$

If C and D are concepts, then $C \sqsubseteq D$ and $C \equiv D$ are *terminological axioms*. The former axioms are called *inclusions* and the latter *equalities*. We can define the semantics of terminological axioms in a straightforward way. An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and it satisfies the equality $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. $\mathcal{I}$ satisfies a set of terminological axioms if it satisfies all axioms in the set. An interpretation, which satisfies a (set of) terminological axioms is called a *model* of this (set of) axioms. Two (sets of) axioms are *equivalent* if they have the same models. An equality whose left hand side is an atomic concept is a *concept definition*. A finite set $\mathcal{T}$ of terminological axioms is called a *(general) TBox*. Let N_I be the set of object names (disjoint with N_R and N_C). An *assertion* has the form $C(a)$ (*concept assertion*) or $r(a, b)$ (*role assertion*), where a, b are object names, C is a concept, and r is a role. An *ABox* $\mathcal{A}$ is a finite set of assertions.

Objects are also called individuals. To allow interpreting ABoxes we extend the definition of an interpretation. Additionally to mapping concepts to subsets of our domain and roles to binary relations, an interpretation has to assign to each individual name $a \in N_I$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all $C(a) \in \mathcal{A}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $r(a, b) \in \mathcal{A}$. An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (written $\mathcal{I} \models \mathcal{K}$) iff it is a model of $\mathcal{T}$ and $\mathcal{A}$.

An $\mathcal{ALC}$ concept is in *negation normal form* if negation only occurs in front of concept names.

As we have seen a knowledge base can be used to store the information we have about the application domain. Besides this *explicit* knowledge we can also deduce *implicit* knowledge from a knowledge base. It is the aim of *inference algorithms* to extract such implicit knowledge. There are some standard reasoning tasks in Description Logics, which we will briefly describe.

In *terminological reasoning* we reason about concepts. The standard problems are *satisfiability* and *subsumption*. Intuitively, satisfiability determines if a concept can be satisified, i.e. it is free of contradictions. Subsumption of two concepts detects if one of the concepts is more general than the other.

Definition 3 (satisfiability). *Let C be a concept and $\mathcal{T}$ a TBox. C is* satisfiable *iff there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. C is* satisfiable with respect to $\mathcal{T}$ *iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.*

Definition 4 (subsumption, equivalence). *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 $C \equiv D$ ($C \equiv_{\mathcal{T}} D$).*

In *assertional reasoning* we reason about objects. The *consistency problem* is the question whether an ABox has a model. The *instance problem* is to find

out whether an object is an instance of a concept, i.e. belongs to it. *Retrieval* is the problem of finding all instances of a given concept.

Definition 5 (consistency). *An Abox $\mathcal{A}$ is* consistent *iff it has a model. An ABox A is* consistent with respect to a TBox $\mathcal{T}$ *iff $\mathcal{A}$ and $\mathcal{T}$ have a common model.*

Definition 6 (instance). *Let $\mathcal{A}$ be an ABox, $\mathcal{T}$ a TBox, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base, C a concept, and $a \in N_I$ an object.* a is an instance of C with respect to $\mathcal{A}$ *, denoted by $\mathcal{A} \models C(a)$, iff in any model $\mathcal{I}$ of $\mathcal{A}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.* a is an instance of C with respect to $\mathcal{K}$ *, denoted by $\mathcal{K} \models C(a)$, iff in any model $\mathcal{I}$ of $\mathcal{K}$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.*

To denote that a is not an instance of C with respect to $\mathcal{A}$ ($\mathcal{K}$) we write $\mathcal{A} \not\models C(a)$ ($\mathcal{K} \not\models C(a)$).

We use the same notation for sets of assertions of the form $C(a)$, e.g. $\mathcal{K} \models S$ means that every element in S follows from $\mathcal{K}$.

In the area of Description Logics a variety of different Description Languages has evolved, which are distinguished mainly by the language primitives they allow for combining concepts and roles. We avoid to go into too much details on this although we will sometimes draw on such knowledge. The interested reader is referred to Baader et al. (2003) for further background.

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 7 (learning problem in Description Logics). *Let a concept name `Target`, a knowledge base $\mathcal{K}$, 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` $\equiv C$ is an acyclic definition and for $\mathcal{K}' = \mathcal{K} \cup \{\texttt{Target} \equiv C\}$ we have $\mathcal{K}' \models E^+$ and $\mathcal{K}' \not\models E^-$.*

When we speak about concepts as possible problem solutions it is useful to introduce some shortcuts for the two main criteria: covering all positive examples and not covering negative examples.

Definition 8 (complete, consistent, correct). *Let C be a concept, $\mathcal{K}$ the background knowledge base, $\texttt{Target}$ the target concept, $\mathcal{K}' = \mathcal{K} \cup \{\texttt{Target} \equiv C\}$ the extended knowledge base, and E^+ and E^- the positive and negative examples.*

C is complete *with respect to E^+ if for any $e \in E^+$ we have $\mathcal{K}' \models e$. C is* consistent *with respect to E^- if for any $e \in E^-$ we have $\mathcal{K}' \not\models e$. C is* correct *with respect to E^+ and E^- if C is complete with respect to E^+ and consistent with respect to 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 (Nienhuys-Cheng and de Wolf, 1997), where refinement operators are widely used to find hypotheses. Intuitively downward (upward) refinement operators construct specialisations (generalisations) of hypotheses.

Definition 9 (refinement operator). *A* quasi-ordering *is a reflexive and transitive relation. In a 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 10 ($\mathcal{L}$ refinement operator). *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 11 (refinement chain). *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 12 (downward and upward cover). *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 Definition 4). 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 13 (weak syntactic equality). *We say that the concepts C and D are* weakly (syntactically) equal*, denoted by $C \simeq D$ iff one of the following conditions holds:*

- $C = \top$ *and* $D = \top$
- $C = \bot$ *and* $D = \bot$
- $C = A$ *and* $D = A$ *($A \in N_C$)*
- $C = \neg C'$ *and* $D = \neg D'$ *and* $C' \simeq D'$
- $C = \exists r.C'$ *and* $D = \exists r.D'$ *and* $C' \simeq D'$
- $C = \forall r.C'$ *and* $D = \forall r.D'$ *and* $C' \simeq D'$
- $C = C_1 \sqcap \cdots \sqcap C_n$ *and* $D = D_1 \sqcap \cdots \sqcap D_n$ *and there is a permutation* $\psi : N \mapsto N$ *with* $N = \{1, \ldots, n\}$ *such that for all* $i \in N$ *we have* $C_i \simeq D_{\psi(i)}$
- $C = C_1 \sqcup \cdots \sqcup C_n$ *and* $D = D_1 \sqcup \cdots \sqcup D_n$ *and there is a permutation* $\psi : N \mapsto N$ *with* $N = \{1, \ldots, n\}$ *such that for all* $i \in N$ *we have* $C_i \simeq D_{\psi(i)}$

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 hypothesis.

Definition 14 (properties of DL refinement operators). *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 $\sqsubseteq$.*

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

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

4 Analysing the Properties of Refinement Operators

In this section we will analyse the properties of refinement operators in Description Logics. The need for such an analysis was expressed in (Esposito et al., 2004, section 5). 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 (Esposito et al., 2004; Iannone and Palmisano, 2005). 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. Some properties for $\mathcal{ALER}$ refinement operators were shown in (Badea and Nienhuys-Cheng, 2000).

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, the next proposition shows that downward covers do exist.

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

Proof. Let $N_R = \{r\}$ and $N_C = \{A\}$. We want to show that $C = \exists r.\top \sqcup A$ is a downward cover of $\top$. We have to show that there is no concept D with $C \sqsubset D \sqsubset \top$.

We will prove that such a concept cannot exist from a semantical point of view. By contradiction, we assume that we have found such a concept D.

Since C is strictly subsumed by D, there is an interpretation $\mathcal{I}_1$ and an object a_1 such that $a_1^{\mathcal{I}_1} \notin C^{\mathcal{I}}$ and $a_1^{\mathcal{I}_1} \in D^{\mathcal{I}_1}$. a_1 cannot have an r-filler, because then we would immediately get $a_1^{\mathcal{I}_1} \in C^{\mathcal{I}} = (\exists r.\top \sqcup A)^I$. Similarly we get that $a_1^{\mathcal{I}_1} \notin A^{\mathcal{I}_1}$.

Since D is strictly subsumed by $\top$, there is an interpretation $\mathcal{I}_2$ and an object a_2 such that $a_2^{\mathcal{I}_2} \notin D^{\mathcal{I}_2}$. Because of $C \sqsubset D$, we know $a_2^{\mathcal{I}_2} \notin C^{\mathcal{I}_2}$. By the same arguments as above we can deduce that a_2 does not have an r-filler in $\mathcal{I}_2$ and $a_2^{\mathcal{I}_2} \notin A^{\mathcal{I}_2}$.

In both interpretations $\mathcal{I}_j$ ($j \in \{1, 2\}$), the associated objects a_j do not have an r-filler and do not belong to A. Additionally there are no other concept names and roles, so both interpretations have to interpret concepts - in particular D - in the same way with respect to the associated objects, i.e. we either have ($a_1^{\mathcal{I}_1} \in D^{\mathcal{I}_1}$ and $a_2^{\mathcal{I}_2} \in D^{\mathcal{I}_2}$) or we have ($a_1^{\mathcal{I}_1} \notin D^{\mathcal{I}_1}$ and $a_2^{\mathcal{I}_2} \notin D^{\mathcal{I}_2}$). (The reason is that to determine, whether $a^{\mathcal{I}} \in E^{\mathcal{I}}$ holds for an arbitrary $\mathcal{ALC}$ concept E and an interpretation $\mathcal{I}$, such that a does not have a role filler in $\mathcal{I}$ and does not belong to any atomic concept, it is not important how objects different from a are interpreted by $\mathcal{I}$ – an interpretation is determined by the values it assigns to role and concept names.) This is a contradiction, because we assumed $a_1^{\mathcal{I}_1} \in D^{\mathcal{I}_1}$ and $a_2^{\mathcal{I}_2} \notin D^{\mathcal{I}_2}$.

Upward covers can be handled analogously, i.e. $\forall r.\bot \sqcap A$ is an upward cover of $\bot$.

The idea in the proof of Proposition 1 can be extended to situations with more than one role and concept name. In this case we obtain the following concept as a downward cover of $\top$ (we do not prove this explicitly, because we do not use this result later on):

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

The result shows 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 in (Badea and Nienhuys-Cheng, 2000). 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 (minimality and weak completeness). *There exists no minimal and weakly complete $\mathcal{AL}$ downward refinement operator.*

Proof. We will look at the case of $N_R = \{r\}$, and $N_C = \emptyset$. To prove the result, it is of course sufficient to look at this special case. With some modifications we can also show the results for the more general case of arbitrarily many atomic concepts and at least one role. However, this would render the proof less readable from a notational point of view.

In the following, let $\mathrm{qd}(C)$ denote the quantor depth of a concept C.

By contradiction, we assume we have a minimal and weakly complete $\mathcal{AL}$ downward refinement operator ρ. Since ρ is weakly complete, it must allow a minimal refinement step $C \rightsquigarrow_\rho D$, such that C does not (syntactically) contain $\bot$ and D (syntactically) contains $\bot$. We will now define a concept C' equivalent to C and a concept D' equivalent to D.

Using the equivalence $\forall r.(C_1 \sqcap C_2) \equiv \forall r.C_1 \sqcap \forall r.C_2$ for all concepts C_1, C_2 and roles r, we can rewrite D to an equivalent concept D', which does not contain parentheses, i.e. D' is of the form $D'_1 \sqcap \dots \sqcap D'_m$, where D'_i does not contain $\sqcap$ symbols for all i $(1 \leq i \leq m)$.

Due the equivalences $\forall r.\top \equiv \top$, $\forall r.(C_1 \sqcap C_2) \equiv \forall r.C_1 \sqcap \forall r.C_2$ (mentioned above), and $C_1 \sqcap \top \equiv C_1$ for all concepts C_1, C_2 and roles r, we can obtain a concept C' by exhaustively replacing the left hand side of the equivalences by their right hand side. Since C, and therefore C', do not contain $\bot$ symbols, we either have $C' = \top$ or C' is of the following form, where S is a list of numbers:

$$C' = \prod_{i=1}^{b} \underbrace{\forall r. \dots \forall r}_{S_a - \text{times}} . \exists r.\top \quad \text{with } S = [s_1, \dots, s_b]$$

This means, that either $C' = \top$ or C' is a conjunction of elements of the form $\forall r. \dots \forall r. \exists r.\top$.

Since $D' = D'_1 \sqcap \dots \sqcap D'_m$ contains a $\bot$ symbol (no $\bot$ symbols are lost during the transformation), there is a j $(1 \leq j \leq m)$ such that D'_j contains a $\bot$ symbol. Let $D'' = C' \sqcap D'_j$. We have $D' \sqsubseteq D''$, due to $D' \equiv D' \sqcap C'$ ($D \equiv D'$ is a downward refinement of $C \equiv C'$) and $D'_j \sqsubseteq D'$. D'_j is of the form $\forall r. \dots \forall r. \bot$, because D'_j contains a $\bot$ symbol and no $\sqcap$ symbol.

We define a new concept E of the following form, where n is an arbitrary number larger than the quantor depths of C' and D'_j:

$$E = C' \sqcap \underbrace{\forall r. \dots \forall r}_{n - \text{times}} . \exists r.\top \quad \text{with } n > \mathrm{qd}(C) \text{ and } n > \mathrm{qd}(D_j)$$

We claim the following:

$$D \equiv D' \sqsubseteq D'' \sqsubset E \sqsubset C' \equiv C$$

If the claim is true, then the refinement step $C \rightsquigarrow_\rho D$ is not minimal, since there exists a concept E strictly more specific than C and strictly more general than D, which finishes the proof.

We have already shown $D' \sqsubseteq D''$, so it remains to show $D'' \sqsubset E$ and $E \sqsubset C'$.

1. $E \sqsubset C'$:
   $E = C' \sqcap \forall r. \dots \forall r. \exists r.\top \sqsubseteq C'$ is obvious, so it remains to show that E and C' are not equivalent.
   We define an interpretation $\mathcal{I}$ with $r^{\mathcal{I}} = \{r(a,b) \mid a = a_i, b = a_{i+1}, 0 \leq i < n\}$, illustrated by:
   $$a_0 \xrightarrow{r} a_1 \xrightarrow{r} \dots \xrightarrow{r} a_n$$

We claim $a_0^{\mathcal{I}} \in C'^{\mathcal{I}}$: If $C' = \top$, then this $E \sqsubset C'$ obvious. Otherwise, C' is a conjunction of elements of the form $\forall r. \ldots \forall r.\exists r.\top$ with quantifier depth smaller than n. For these elements, we have $a_0^{\mathcal{I}} \in (\forall r. \ldots \forall r.\exists r.\top)^{\mathcal{I}}$ and therefore $a_0^{\mathcal{I}} \in C'^{\mathcal{I}}$.
For E we have $a_0^{\mathcal{I}} \notin E^{\mathcal{I}}$, because a_n does not habe an r-successor. Thus, C' and E are not equivalent.

2. $D'' \sqsubset E$:
We will first show $D'' \sqsubseteq E$:

$$D'' = C' \sqcap \forall r. \ldots \forall r.\bot \sqsubseteq C' \sqcap \forall r. \ldots \forall r.\exists r.\top = E$$

We have $\mathrm{qd}(D'') + 1 < \mathrm{qd}(E)$, hence D'' cannot be more general than E.
To show that D'' and E are not equivalent, we define an interpretation $\mathcal{I}$ with $r^{\mathcal{I}} = \{r(a, a)\}$. We have $a^{\mathcal{I}} \in E^{\mathcal{I}}$, because $a^{\mathcal{I}} \in (\forall r. \ldots \forall r.\exists r.\top)^{\mathcal{I}}$ holds for all concepts of the form $\forall r. \ldots \forall r.\exists r.\top$.
We have $a^{\mathcal{I}} \notin D'^{\mathcal{I}}$, because $a^{\mathcal{I}} \notin (\forall r. \ldots \forall r.\bot)^{\mathcal{I}}$ for all concepts of the form $\forall r. \ldots \forall r.\bot$.
Hence, D'' and E are not equivalent.

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

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

Proof. Consider the downward refinement operator ρ defined by

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

where $|D|$ stands for the number of symbols in D. The operator can do one of two things:

- add a $\top$ symbol
- generate the set of all concepts up to a certain length, which are subsumed by C

The operator is finite, because the set of all concepts up to a given length is finite (and the singleton set $\{C \sqcap \top\}$ is finite).

The operator is complete, because given a concept C we can reach an arbitrary concept D with $D \sqsubset C$. This is obvious, because we only need to add $\top$-symbols until there are $|D|$ occurences of $\top$. Within the next step we can then be sure to reach D.

For upward refinement operators we can use an analogous operator ϕ, which is also complete and finite:

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

Remark 1 (complete and finite refinement operators). In (Badea and Nienhuys-Cheng, 2000) it was stated that there can be no complete and finite $\mathcal{ALER}$ refinement operator. However, this is not correct, because by using the technique in the proof of Proposition 3 one can construct a finite and complete $\mathcal{ALER}$ refinement operator.

Of course, it is obvious that the operator used to prove Proposition 3 is not useful in practice, since it merely generates concepts without paying attention to efficiency. However, we are interested in theoretical limits of refinement operators, so it is a valid method to consider impractical operators. It is indeed 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 (in the operator this was done by adding $\top$ symbols). 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 indiciates that learning concepts in sufficiently expressive Description Logics is hard.

Proposition 4 (ideal refinement operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, there does not exist any ideal $\mathcal{L}$ refinement operator.*

Proof. By contradiction, we assume that there exists an ideal downward refinement operator ρ. We further assume that there is a role $r \in N_R$. Let $\rho(\top) = \{C_1, \ldots, C_n\}$ be a set of refinements of the $\top$ concept. (This set has to be finite.) Let m be a natural number larger than the maximum of the *quantor depths* (depth of the nesting of quantifications) of the concepts in $\rho(\top)$. We construct a concept D as follows:

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

What is the meaning of D? As we can see, the semantics of $\forall r.\bot$ is that an object must not have an r-filler and the semantics of $\exists r.\top$ is that an object must have an r-filler. We say that an object $a \in N_I$ has an *r-successor at distance n in $\mathcal{I}$* if there is a set of objects $\{a_0, \ldots, a_n\} \subseteq N_I$ with $a = a_0, (a_0^{\mathcal{I}}, a_1^{\mathcal{I}}) \in r^{\mathcal{I}}, \ldots, (a_{n-1}^{\mathcal{I}}, a_n^{\mathcal{I}}) \in r^{\mathcal{I}}$. The meaning of D is that an object does not have an r-successor at distance m in $\mathcal{I}$ or it has an r-successor at distance $m+1$ in $\mathcal{I}$. Formally for an arbitrary object a and an interpretation $\mathcal{I}$ we have $a^{\mathcal{I}} \in D^{\mathcal{I}}$ iff a does not have an r-successor at distance m in $\mathcal{I}$ or a has an r-successor at distance $m+1$ in $\mathcal{I}$.

D is not equivalent to $\top$. For the interpretation $\mathcal{I}$ with $r^{\mathcal{I}} = \{r(a,b) \mid a = a_i, b = a_{i+1}, 0 \leq i < m\}$, illustrated by

$$a_0 \xrightarrow{r} a_1 \xrightarrow{r} \ldots \xrightarrow{r} a_m$$

we have $a_0^{\mathcal{I}} \notin D^{\mathcal{I}}$.

As a prerequisite for proving the proposition, we want to show that there exists no concept with a quantifier depth smaller than m, which strictly subsumes

D and is not equivalent to $\top$. By contradiction, we assume such a concept E with $D \sqsubset E \sqsubset \top$ exists. Since we have $E \not\equiv \top$, there exists an interpretation $\mathcal{I}$ and an object a such that $a^{\mathcal{I}} \notin E^{\mathcal{I}}$. By $D \sqsubset E$, this also implies $a^{\mathcal{I}} \notin D^{\mathcal{I}}$. We do a case distinction on a and $\mathcal{I}$:

1. a does not have an r-successor at distance m in $\mathcal{I}$: By the semantics of D, as discussed above, this means $a^{\mathcal{I}} \in D^{\mathcal{I}}$, which contradicts $a^{\mathcal{I}} \notin D^{\mathcal{I}}$.
2. a has an r-successor at distance m in $\mathcal{I}$:
   We can view the interpretation $\mathcal{I}$ as a directed graph in a straightforward way. The set of nodes is $\{b^{\mathcal{I}} \mid b \in N_I\}$ and the edges are defined by $\{(b, c) \mid (b, c) \in r^{\mathcal{I}}\}$. This graph does not contain a cycle, which is reachable from $a^{\mathcal{I}}$, because then we would have $a^{\mathcal{I}} \in D^{\mathcal{I}}$ due to the $\exists r. \ldots . \exists r. \top$ part of D (if we have a reachable cycle, there always exists a successor). Hence, $a^{\mathcal{I}}$ spans an acyclic graph, i.e. a tree, of successors in $\mathcal{I}$.
   Because of $a^{\mathcal{I}} \notin D^{\mathcal{I}}$, we know that there is a path of length m starting from $a^{\mathcal{I}}$ in this tree (due to the $\forall r. \ldots . \forall r. \bot$ part of D). This means we have pairwise different objects $a_0, \ldots, a_m$ such that $a = a_0$ and $(a_0^{\mathcal{I}}, a_1^{\mathcal{I}}) \in r^{\mathcal{I}}, \ldots, (a_{m-1}^{\mathcal{I}}, a_m^{\mathcal{I}}) \in r^{\mathcal{I}}$.
   We create a new interpretation $\mathcal{I}'$ from $\mathcal{I}$ by adding a new object a_{m+1} and changing $r^{\mathcal{I}}$ to $r^{\mathcal{I}'} = r^{\mathcal{I}} \cup \{(a_m^{\mathcal{I}}, a_{m+1}^{\mathcal{I}})\}$. Since E has a quantifier depth smaller than m, we know that a_{m+1} is out of the scope of these quantifiers, so we can deduce[4] $a^{\mathcal{I}'} \notin E^{\mathcal{I}'}$ from $a^{\mathcal{I}} \notin E^{\mathcal{I}}$. However, a has a successor at distance m in $\mathcal{I}'$, so by the semantics of D we have $a^{\mathcal{I}'} \in D^{\mathcal{I}'}$. This implies $a^{\mathcal{I}'} \in E^{\mathcal{I}'}$ due to $D \sqsubset E$, which is contradiction to $a^{\mathcal{I}'} \notin E^{\mathcal{I}'}$.

Hence we have shown that there does not exist a more general concept than D, which is not equal to $\top$, with a quantifier depth smaller than m. 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.

Proposition 5 (complete and proper refinement operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, there exists a complete and proper $\mathcal{L}$ refinement operator.*

Proof. The proof is trivial, because we can just use $\rho(C) = \{D \mid D \sqsubset C\}$ as downward refinement operator, which is obviously complete and proper. For upward refinement we can analogously consider $\rho(C) = \{D \mid C \sqsubset D\}$.

We have shown that the combination of completeness and properness is possible. Propositions 3, 4, and 5 state that for complete refinement operators, which are usually desirable, one has to sacrifice properness or finiteness. We will now look at non-redundancy.

[4] This argument hinges on the language used. It covers $\mathcal{ALC}$ and all other well-known quantifiers such as number restrictions, but it may fail for exotic extensions (when a quantifier has a larger scope than only the role fillers of an object.

Proposition 6 (complete, non-redundant refinement operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, 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 Proposition 5, 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:

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

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 fom 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. Hence, ρ' is non-redundant.

Proposition 7 (complete, non-redundant refinement operators II). *Let ρ be an arbitrary $\mathcal{L}$ refinement operator, where $\mathcal{L}$ satisfies the conditions stated in Definition 10, 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 mild: If we have $\bot \in \rho^*(C)$ for any concept C, then it is already fulfilled. 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. We will prove an even more general result now.

Proof. Let ρ be a complete downward refinement operator, C_{up}, C_{down} be concepts with $C_{\text{down}} \sqsubset C_{\text{up}}$, $\{C \mid C \in \rho^*(C_{\text{up}}), C \equiv C_{\text{down}}\}$ be finite, and S be an infinite set of concepts, which are pairwise incomparable, strictly subsumed by C_{up} and strictly subsume C_{down}. For instance, we could have $C_{\text{up}} = \top$, $C_{\text{down}} = \bot$ and $S = \{\forall r.A, \forall r.\forall r.A, \dots\}$.

Due to the completeness of ρ, there must be a refinement chain from each concept in S to a concept, which is equivalent to C_{down}. Since there are infinitely many concepts in S, but only finitely many different syntactic representations of C_{down} are reached, there have to be concepts C'_{down}, C_1, and C_2, such that $C'_{\text{down}} \in \rho^*(C_1)$ and $C'_{\text{down}} \in \rho^*(C_2)$.

Because of $C_1 \not\sqsubseteq C_2$ and $C_2 \not\sqsubseteq C_1$, we know $C_1 \notin \rho^*(C_2)$ and $C_2 \notin \rho^*(C_1)$. Hence, there exists a refinement chain from C_{up} to C_{down} through C_1 and a refinement chain from C_{up} to C_{down}, which goes through C_2 and not through C_1. Thus, ρ is redundant.

Again, the proof for upward refinement operators is analogous.

As a consequence, completeness and non-redundancy cannot be combined. Usually it is 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 (incomplete refinement operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, 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 (properties of $\mathcal{L}$ refinement operators (I)). *Let $\mathcal{L}$ be a language, which satisfies the conditions stated in Definition 10. 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.

Proof. The theorem is a consequence of the previous results. We have seen that downward and upward operators allow the same combinations of properties, so it is not necessary to distinguish between them. To be sure to cover all combinations of properties we make a case distinction.

1. The operator is complete. In this case we cannot add non-redundancy (Proposition 7). Finiteness (Proposition 3) and properness (Proposition 5) can be added separately, but not both at the same time (Proposition 4).
2. The operator is not complete. In this case we can add all other properties (Proposition 8).

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. As a first observation, we see that the arguments in the proof of Proposition 7, which have shown that an $\mathcal{L}$ refinement operator cannot be complete and non-redundant, do no longer apply if we consider weak completeness and non-redundancy.

Proposition 9 (weakly complete, non-redundant, and proper operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, there exists a weakly complete, non-redundant, and proper $\mathcal{L}$ refinement operator.*

Proof. The following operator is weakly complete, non-redundant, and proper:

Let S be a maximal subset of $\{C \mid C \not\equiv \top\}$ with $C_1, C_2 \in S \implies C_1 \not\sqsupseteq C_2$.

$$\rho(C) = \begin{cases} S & \text{if } C = \top \\ \emptyset & \text{otherwise} \end{cases}$$

Such a set S as used in the definition of the operator indeed exists. It contains one representative of each equivalence class with respect to weak equality of the set $\{C \mid C \not\equiv \top\}$. The operator is proper, since it contains only mappings of the top concept to concepts, which are not equivalent to top. It is non-redundant, because there is no refinement chain of length greater than 1 and all concepts we reach are pairwise not weakly equal. It is weakly complete, because for every concept, which is not equivalent to $\top$, we can reach an equivalent concept from $\top$ by ρ.

The corresponding upward refinement operator is: Let S be a maximal subset of $\{C \mid C \not\equiv \bot\}$ with $C_1, C_2 \in S \implies C_1 \not\geq C_2$.

$$\rho(C) = \begin{cases} S & \text{if } C = \bot \\ \emptyset & \text{otherwise} \end{cases}$$

The operator just given is obviously not useful in practice, but it suffices for the proof of the proposition.

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

Proof. The following operator is weakly complete, non-redundant, and finite:

For an arbitrary concept C, let S_C be a maximal subset of $\{D \mid D \sqsubset \top \text{ and } |D| = \text{number of } \top \text{ occurences in } C\}$ with $C_1, C_2 \in S_C \implies C_1 \not\geq C_2$.

$$\rho(C) = \begin{cases} \{\underbrace{\top \sqcap \cdots \sqcap \top}_{n+1 \text{ times } \top}\} \cup S_C & \text{if } C = \underbrace{\top \sqcap \cdots \sqcap \top}_{n \text{ times } \top} \\ \emptyset & \text{otherwise} \end{cases}$$

The operator is finite, because S_C is finite for any concept C (the number of concepts with a fixed length is finite). It is weakly complete, because every concept C with $C \sqsubset \top$ can be reached from $\top$. This is done by accumulating $\top$ symbols until we have $|C|$ such symbols and then generate C. The operator is furthermore non-redundant, because obviously for any concept there is exactly one path for reaching the concept via iterated applications of ρ to $\top$.

The corresponding upward operator is analogous. It works by accumulating $\bot$ symbols instead of $\top$ symbols and generates concepts, which are strictly more general than $\bot$.

Corollary 1 (weakly complete, proper, and finite operators). *For any language $\mathcal{L}$, which satisfies the conditions stated in Definition 10, there exists no weakly complete, finite, and proper $\mathcal{L}$ refinement operator.*

Proof. To show this we can use the proof of Proposition 4. There we have shown that in a finite and proper $\mathcal{L}$ refinement operator there exists a concept, which cannot be reached from the $\top$ concept. This means that such an operator cannot be weakly complete.

The result of the previous observations is that, when requiring only weak completeness instead of completeness, non-redundant operators are possible. The following theorem is the result of the full analysis of the desired properties of $\mathcal{L}$ refinement operators.

Theorem 2 (properties of refinement operators (II)). *Let $\mathcal{L}$ be a language, which satisfies the conditions stated in Definition 10. 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.

Proof. We can do a similar case distinction as in Theorem 1. The first case (complete operator) is analogous except that obviously a complete operator is also weakly complete. For the second case (operator is not complete) we can make a simple case distinction again:

1. The operator is weakly complete. Propositions 9 and 10 have shown that weakly complete operators can be non-redundant and proper as well as non-redundant and finite. Proposition 1 shows that finiteness and properness cannot be combined, so these sets of properties are maximal.
2. The operator is not weakly complete. In this case we can add all remaining properties (Proposition 8), i.e. non-redundany, finiteness, and properness.

Remark 2 (subsumption with respect to a TBox). 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. All negative results, i.e Propositions 4, 7, Corollary 1, we can consider subsumption with respect to an empty TBox as example. In all positive results, i.e. Propositions 3, 5, 8, 9, 10, we can rewrite the example operators by replacing $\sqsubseteq$ with $\sqsubseteq_{\mathcal{T}}$.

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 (Badea and Nienhuys-Cheng, 2000) 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. In (Esposito et al., 2004; Iannone and Palmisano, 2005) 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. (Esposito et al., 2004) stated that an investigation of the properties of refinement operators in Description Logics, as we have done in this article, is required. In (Fanizzi et al., 2004) downward refinement for $\mathcal{ALN}$ was analysed using a clausal representation of DL concepts. Refinement operators have also been dealt with in hybrid systems. In (Lisi and Malerba, 2003) 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 (Kietz and Morik, 1994; Cohen and Hirsh, 1994) learning algorithms for Description Logics without refinement operators were analysed.

In the area of Inductive Logic Programming (Nienhuys-Cheng and de Wolf, 1997) considerable efforts have been made to analyse the properties of refinement operators. In general using refinement operators for clauses to learn in Description Logics is usually not a good choice (Badea and Nienhuys-Cheng, 2000). However, the theoretical foundations of refinement operators in general also apply to Description Logics, which is why we want to mention work in this area here.

A milestone in Machine Learning (Mitchell, 1997) in general was the Model Inference System in (Shapiro, 1991). Shapiro describes how refinement operators can be used to adapt a hypothesis to a sequence of examples. Afterwards, refinement operators became widely used as a learning method. (van der Laag and Nienhuys-Cheng, 1994) 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 (van der Laag and Nienhuys-Cheng, 1994). Unfortunately, we could not make use of these results directly, because proving properties of covers in Description Logics without using a specific language is likely to be harder than directly proving the results.

(Nienhuys-Cheng et al., 1993) discussed refinement for different versions of subsumption, in particular weakenings of logical implication. In (Nienhuys-Cheng et al., 1999) 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 (Badea and Stanciu, 1999). Since such operators do not exist for clauses ordered by θ-subsumption (van der Laag and Nienhuys-Cheng, 1994), weaker versions of subsumption were considered. This was later extended to theories, i.e. sets of clauses (Fanizzi et al., 2003). A less widely used property of refinement operators, called flexibility, was discussed in (Badea, 2000). Flexibility essentially means that previous refinements of an operators 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.

Further work will focus on leveraging the theoretical insights for defining practical refinement operators. On the long-term, we intend to develop a learning system for description logics and to evaluate it in realistic scenarios.

Bibliography

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Badea, L. (2000). Perfect refinement operators can be flexible. In Horn, W., editor, *Proceedings of the 14th European Conference on Artificial Intelligence*, pages 266–270. IOS Press.

Badea, L. and Nienhuys-Cheng, S.-H. (2000). A refinement operator for description logics. In Cussens, J. and Frisch, A., editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 40–59. Springer-Verlag.

Badea, L. and Stanciu, M. (1999). Refinement operators can be (weakly) perfect. In Džeroski, S. and Flach, P., editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer-Verlag.

Brachman, R. J. (1978). A structural paradigm for representing knowledge. Technical Report BBN Report 3605, Bolt, Beraneck and Newman, Inc., Cambridge, MA.

Cohen, W. W. and Hirsh, H. (1994). Learning the classic description logic: Theoretical and experimental results. In Jon Doyle, Erik Sandewall, P. T., editor, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 121–133, Bonn, FRG. Morgan Kaufmann.

Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., and Semeraro, G. (2004). 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.

Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I., and Semeraro, G. (2004). 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.

Fanizzi, N., Ferilli, S., Mauro, N. D., and Basile, T. M. A. (2003). Spaces of theories with ideal refinement operators. In Gottlob, G. and Walsh, T., editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 527–532. Morgan Kaufmann.

Iannone, L. and Palmisano, I. (2005). An algorithm based on counterfactuals for concept learning in the semantic web. In Ali, M. and Esposito, F., editors, *Innovations in Applied Artificial Intelligence*, pages 370–379, Bari, Italy. Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.

Kietz, J.-U. and Morik, K. (1994). A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14:193–217. Re-

vised and extended version of a paper presented at Workshop W8 of the 12th IJCAI-91: Evaluating and changing representations in machine learning. Also as Arbeitspapiere der GMD No. 716.

Lisi, F. A. and Malerba, D. (2003). Ideal refinement of descriptions in AL-log. In Horváth, T., 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.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill, New York.

Nienhuys-Cheng, S.-H. and de Wolf, R., editors (1997). *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer.

Nienhuys-Cheng, S.-H., Laer, W. V., Ramon, J., and Raedt, L. D. (1999). Generalizing refinement operators to learn prenex conjunctive normal forms. In Džeroski, S. and Flach, P., editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 245–256. Springer-Verlag.

Nienhuys-Cheng, S. H., van der Laag, P. R. J., and van der Torre, L. W. N. (1993). Constructing refinement operators by decomposing logical implication. In Torasso, P., 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. Springer.

Shapiro, E. Y. (1991). Inductive inference of theories from facts. In Lassez, J. L. and Plotkin, G. D., editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 199–255. The MIT Press.

van der Laag, P. R. J. and Nienhuys-Cheng, S.-H. (1994). Existence and nonexistence of complete refinement operators. In Bergadano, F. and Raedt, L. D., editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 307–322. Springer-Verlag.