Optimal Interpolation in $\mathcal{ALC}$

Stefan Schlobach

Language and Inference Technology,
ILLC, Universiteit van Amsterdam, NL,
Email: schlobac@science.uva.nl

Abstract

We describe ongoing research to support the construction of terminologies with Description Logics. Both in *explanation of subsumption* and in *learning of terminologies* we search for particular concepts because of their syntactic and semantic properties. More precisely, the set of explanations for a subsumption $P \sqsubseteq N$ is the set of *optimal interpolants* for P and N, and similarly for the learning application. We provide definitions for optimal interpolation and an algorithm based on Boolean minimisation of concept-names in a tableau proof for $\mathcal{ALC}$-satisfiability.

1 Introduction

Building ontologies is a time consuming and error-prone process and one of the bottle-necks in a number of AI applications. We have recently suggested various methods to tackle this problem. First, we investigated methods to automatically learn terminologies from data [14] and, secondly, we added explanation facilities to terminological reasoning [16]. Modern Description Logic (DL, see Chapter 1 and 2 of [1] for a detailed introduction) reasoning systems efficiently answer queries, e.g., whether a concept $\exists child.\top \sqcap \forall child.Doctor$ is *subsumed* by $\exists child.(Doctor \sqcup Rich)$, i.e. whether one is a subclass of the other. Unfortunately, they do not provide explanatory information. In our approach a suitable explanation for this subsumption is that the former concept is more special than a third concept $\exists child.Doctor$ which, again, is more special than the latter. Not only is this concept, which we will call an *illustration*, strongly linked by the common vocabulary to both the subsumer and the subsumed concept, it is also the simplest such illustration.

A traditional approach to supervised learning of terminological axioms is to iteratively build least common subsumers (lcs) summarising the properties of as many examples as possible, making sure that none of the counterexamples is covered (e.g. [4]). Our approach has a more "inductive" flavour as we search for generalisations of the examples which are interesting because of their syntactic properties. Let us illustrate the general idea with a toy exam-

ple. Suppose we try to learn the definition of a concept *Rabbit* from an example *rab:LongEars* $\sqcap$ *Fur* $\sqcap$ *Paw* and counterexamples *croco:*$\neg$*LongEars* $\sqcap$ $\neg$*Fur* $\sqcap$ *Paw* and *seal:*$\neg$*LongEars* $\sqcap$ *Fur* $\sqcap$ $\neg$*Paw*. The result of lcs-learning is now simply *LongEars* $\sqcap$ *Fur* $\sqcap$ *Paw* which means that a previously unknown animal with long ears and fur will *not* be classified as a rabbit unless we find out that it has got paws. An alternative is *top-down* learning: take the negation of the properties of the counterexamples as upper bound of the possible learning concepts. From these concepts, e.g. *LongEars* $\sqcap$ *Fur* or *Fur* $\sqcap$ *Paw*, one then looks for those defining smallest axioms (e.g. *Rabbit*$\sqsubseteq$*LongEars*), or for those built from an non-reducible set of concept-names (e.g. *Rabbit*$\sqsubseteq$*Fur* $\sqcap$ *Paw*). Formally, we look for concepts which are more general than the examples $\mathbf{P}$ and inconsistent with the counterexamples $\mathbf{N}$. This corresponds to the subsumption relation: $\models \bigsqcup_{P\in\mathbf{P}} P \sqsubseteq L_D \sqsubseteq \bigsqcap_{N\in\mathbf{N}} \neg N$ and it remains to choose the most suitable of all L_D as definitions in our target axioms.

The solutions we propose for our applications can be generalised; given two concepts C and D we need to find an *interpolant*, i.e. a concepts I such that $\models C \sqsubseteq I \sqsubseteq D$, and where I is syntactically related to both C and D. Moreover, we need interpolants which are *optimal* with respect to particular syntactic properties such as number of concept-names or size. Algorithms for interpolation for concept subsumption in $\mathcal{ALC}$ are well known (e.g. [13]) as $\mathcal{ALC}$ is a notational variant of modal $\mathbf{K}$. In this paper we extend these algorithms in order to calculate an interpolant I with a minimal number of concept-names, i.e., where there are no interpolants built from a proper subset of the concept-names occurring in I. For this purpose we saturate a tableau according to the usual rules for $\mathcal{ALC}$-satisfiability. Boolean minimisation then renders *reducts*, minimal sets of concept-names preserving the existence of interpolants. Finally, given reducts we calculate optimal interpolants.

The remainder of the paper is organised as follows: In Section 2 we discuss the applications of explanation and concept learning in more detail. Section 3 introduces interpolation for DL, more specifically for $\mathcal{ALC}$, with the vocabulary $\mathcal{L}$ of a concept, and formal notions of optimality. In Section 4 we provide an algorithm to calculate optimal $\mathcal{L}$-interpolants based on Boolean minimisation of concept-names w.r.t. the closure of a tableau for $\mathcal{ALC}$-satisfiability. We finish with an evaluation and a brief discussion of related and further work.

This report is an extended version of a paper presented at M4M-03, Models for Modalities 2003. It contains the relevant proofs in the Appendix A.

2 Applications

Both in *explanation of subsumption* and *learning of terminologies* we require concepts which turn out to be particular interpolants. We only sketch the basic ideas to motivate the need for optimal interpolants deliberately skipping some interesting discussion and alternatives related to the applications.

2.1 Explaining Subsumption by Illustration

Recently, the DL community has shown growing interest in explanation of reasoning (e.g. [6]) to provide additional information to increase the acceptance of logical reasoning, and to give additional insights to the structure of represented knowledge. Let us consider a variant of an example introduced in [3] for the author's "explanation as proof-fragment" strategy, where a concept $C_{ex} := \exists child.\exists child.Rich \sqcap \forall child.\neg((\exists child.\neg Doctor) \sqcup (\exists child.Lawyer))$ is subsumed by $D_{ex} := \exists child.\forall child\ (Rich \sqcup Doctor)$. Instead of providing a concise and simplified extract of a formal proof as an explanation as done in [3] we suggest an alternative, more static approach, which we call *explaining by illustration*. Imagine the above information is given in natural language: *Suppose somebody has a rich grand-child, and each child has neither a child which is not a doctor nor a child which is a Lawyer. Then, this person must have a child, every child of which is either rich or a doctor.* A natural language explanation for this statement is: *The person described above must have a child every child of which is a doctor.* This intermediate statement can be considered an *illustration* of $\models C_{ex} \sqsubseteq D_{ex}$. It can be formalised as $I_{ex} := \exists child.\forall child.Doctor$, and it subsumes C_{ex} and is subsumed by D_{ex}. Moreover, it is constructed from vocabulary both in C_{ex} and D_{ex}, e.g., the information that the person's grandchildren might be a Lawyer is irrelevant as, just from D_{ex}, we don't know anything about grandchildren being Lawyers or not. Finally, the illustration should use a minimal number of concept-names and be of minimal size, as this increases the likelihood of it being understandable.

2.2 Learning Terminologies from Examples

A related problem is the automatic construction of terminologies from examples. Let the following ABox $\mathcal{A}$ describe the experience from which the author of a scientific paper wants to devise a procedure to improve the quality of his new publications, given the knowledge that his previous paper p_1 had been accepted, whereas papers p_2 and p_3 had been rejected before.

$p_1 : \forall statement.(Theorem \sqcup Footnote \rightarrow Interesting) \sqcap \forall statement.\forall implies.(Corr \rightarrow Interesting) \sqcap \exists statement.\exists proof.(Proof \sqcap Understandable) =: P_1$

$p_2 : \exists statement.(Theorem \sqcap \neg Interesting \sqcap \exists implies.(Corr \sqcap \neg Interesting)) =: N_1$

$p_3 : \exists statement.(Theorem \sqcap \neg Interesting) \sqcap \forall statement.\forall proof.(\neg Understandable \sqcup Simple) =: N_2$

From this ABox $\mathcal{A}_P$ the author would like to learn a new TBox axiom such as $Accept \sqsubseteq L_{Accept}$, where $L_{Accept} := \forall statement.(Theorem \rightarrow Interesting)$ is a new concept defining the previously undefined concept-name *Accept*.

Usually, a minimal requirement for supervised learning is that most positive examples are instances of the newly learned concept, but none of the counterexamples. This means that $p_1 \in_{\mathcal{A}} L_{Accept}, p_2 \notin_{\mathcal{A}} L_{Accept}$ and $p_3 \notin_{\mathcal{A}} L_{Accept}$. The first condition corresponds to $\models P_1 \sqsubseteq L_{Accept}$. We strengthen Conditions two

and three as we want a learned axiom to hold even if new information becomes available about the negative examples. Making sure that $N_1 \sqcap L_{Accept} = \bot$ and $N_2 \sqcap L_{Accept} = \bot$ this "exclusiveness" is guarantied and conditions two and three follow. [1] Let us define our approach to *learning of terminologies* in more detail.

Let $\mathbf{P}$ and $\mathbf{N}$ be sets of concepts with positive and negative examples of an new concept D. We say that an axiom $D \sqsubseteq L_D$ is *correct* if $\models P \sqsubseteq L_D$ and $N \sqcap L_D = \bot$ for all $P \in \mathbf{P}$ and $N \in \mathbf{N}$. In machine learning terminology the set of concepts L_D in the corresponding subsumption relation $\models \bigsqcup_{P \in \mathbf{P}} P \sqsubseteq L_D \sqsubseteq \bigsqcap_{N \in \mathbf{N}} \neg N$ determines the *version space* [11]. Finding the appropriate targets depends on the choice of *bias*. Note, that not every correct axiom is an appropriate learning target. The axiom *Accept* $\sqsubseteq \forall$*statement.(Footnote* $\sqcup$ *Theorem* $\rightarrow$*Interesting)* requires all footnotes to be interesting for a paper to be accepted. This, however, cannot be inferred, as there is no information about footnotes in the counterexamples. We need to ensure that new axioms are also syntactically related to examples and counterexamples. In Definition 3.1 we define such a notion of syntactic relation which takes quantifiers and polarity into account, i.e., $\exists r.(C \sqcup \forall s.D)$ is related to $\exists r.\forall s.D$, but not to C or $\exists r.\neg C$. Moreover, following Occam's razor, we assume that simpler axioms are better learning results. The learning targets of concept learning are therefore *optimal axioms* defined as follows; let $\mathcal{N}(C)$ be the set of concept-names occurring in a concept C, $\mathbf{P}$ and $\mathbf{N}$ be the positive and negative examples for a class D. A correct TBox axiom $D \sqsubseteq L_D$ is *optimal* if there is no correct axiom $D \sqsubseteq L'_D$ where L'_D is also syntactically related to $\mathbf{P}$ and $\mathbf{N}$, and where $\mathcal{N}(L'_D) \subset \mathcal{N}(L_D)$.

3 Optimal Interpolation

Calculating illustrations and optimal TBox axioms are interpolation problems, more precisely, problems of finding *optimal interpolants*. Remember that an axiom $D \sqsubseteq L_D$ is correct if $\models \bigsqcup_{P \in \mathbf{P}} P \sqsubseteq L_D \sqsubseteq \bigsqcap_{N \in \mathbf{N}} \neg N$ for examples $\mathbf{P}$ and counterexamples $\mathbf{N}$ of a concept D, and that an illustration for the subsumption $\models C \sqsubseteq D$ was a concept I s.t. $\models C \sqsubseteq I \sqsubseteq D$. In both cases we have argued that common vocabulary and syntactic minimality is desirable.

An *interpolant* for concepts P (for the positive) and N (the negative examples), where $\models P \sqsubseteq N$, is a concept I which is more general than P but more special than N. Furthermore, I has to be built from the vocabulary occurring both in P and N. In the standard definition of interpolation [5] the vocabulary of a formula φ is defined as the set of non-logical symbols in φ. Because we use interpolants for learning and explanation we propose a stronger notion of vocabulary for concepts: including information about the context in which the

[1] In this paper we assume that the *msc* ([10]) exists for each individual, and use sets of concepts describing the properties of individuals. To both sets of concepts and sets of individuals for positive and negative examples we will simply refer to as *examples*.

non-logical symbols (such as concept- and role-names) occur. Formally, $\mathcal{L}(C)$ is a set of pairs $(A,+)^S$ or $(B,-)^S$ of concept-names A, B and polarity $+$ or $-$, labelled with sequences S of role-names. A concept-name A has positive (negative) polarity if it is embedded in an even (odd) number of negations. $\bot$ and $\top$ always occur without polarity (represented by pairs $(\bot, \_)$ and $(\top, \_)$). $\overline{\mathcal{L}}(C)$ denotes the set $\mathcal{L}(C)$ where the polarity of each pair is interchanged (i.e. $+$ replaced by $-$ and vice versa). Furthermore, for a set S, let S^r denote that the sequence of role-names for each element of S has been extended by r. The vocabulary $\mathcal{L}(C)$ of a concept C is then defined as follows.

Definition 3.1 $\mathcal{L}$ is a mapping from $\mathcal{ALC}$ to triples of concept-names, polarities and sequences of role-names defined as follows:

- $\mathcal{L}(\top) = \mathcal{L}(\bot) := \{(\top, \_)^\epsilon, (\bot, \_)^\epsilon\}$
- $\mathcal{L}(A) := (A,+)^\epsilon \cup \mathcal{L}(\top)$ if A is atomic
- $\mathcal{L}(\exists r.C) = \mathcal{L}(\forall r.C) := \mathcal{L}(C)^r \cup \mathcal{L}(\top)^\epsilon$
- $\mathcal{L}(C \sqcap D) := \mathcal{L}(C) \cup \mathcal{L}(D)$
- $\mathcal{L}(C \sqcup D) := \mathcal{L}(C) \cup \mathcal{L}(D)$
- $\mathcal{L}(\neg C) := \overline{\mathcal{L}}(C)$

Take a concept $\forall r.(D \sqcap \neg \exists s.C)$. The related language is the set $\{(C,-)^{rs}$, $(D,+)^r, (\top, \_)^\epsilon, (\top, \_)^r, (\top, \_)^{rs}, (\bot, \_)^\epsilon, (\bot, \_)^r, (\bot, \_)^{rs}\}$. Note that this definition implies that $\mathcal{L}(\bot) \in \mathcal{L}(C)$ and $\mathcal{L}(\top) \in \mathcal{L}(C)$ for every concept C.

The set of interpolants w.r.t. $\mathcal{L}$ will be denoted by $I(P,N)$. In applications where interpolants are used for explanation or learning, additional restrictions are important to identify optimal illustrations or learning targets. There are several types of syntactic restrictions, e.g., interpolants with a minimal set of concept- or role-names or of minimal size, but we will focus on concept-name optimal interpolants. To simplify the presentation we only consider concept interpolation, i.e. interpolation for concept subsumption w.r.t. empty TBoxes.

Definition 3.2 Let P and N be concepts, and let $\mathcal{N}(C)$ denote the set of concept-names occurring in an arbitrary concept C. A concept I is an *optimal interpolant* for P and N if $\models P \sqsubseteq I$ and $\models I \sqsubseteq N$, $\mathcal{L}(I) \subseteq \mathcal{L}(P) \cap \mathcal{L}(N)$, and if there is no interpolant I' for P and N with $\mathcal{N}(I') \subset \mathcal{N}(I)$.

Take $I_{ex} := \exists child.\forall child.Doctor$ from Section 2.1 which is an interpolant for $C_{ex} := \exists child.\exists child.Rich \sqcap \forall child.\neg((\exists child.\neg Doctor) \sqcup (\exists child.Lawyer))$ and $D_{ex} := \exists child.\forall child.(Rich \sqcup Doctor)$, as $\mathcal{L}(I_{ex}) = \{(Doctor,+)^{child\ child}, \ldots\}$ is a subset of $\mathcal{L}(C_{ex}) \cap \mathcal{L}(D_{ex})$. Moreover, the set of concept-names in I_{ex} is $\{Doctor\}$ which is minimal, and I_{ex} is an optimal interpolant for C_{ex} and D_{ex}. Such a minimal set of concept-names will be called a *reduct*.

To define reducts we need some more notation. Let S be an arbitrary subset of the common language of two concepts P and N. The set of interpolants for P and N built from concept-names in S only will be denoted as $I_S(P,N)$. For C_{ex} and D_{ex} and a set $S = \{Doctor, Rich\}$ the set $I_{\{Doctor,Rich\}}(C_{ex}, D_{ex})$ then contains, e.g., $\exists child.\forall child.Doctor$ and $\exists child.\forall child.Doctor \sqcup Rich$. Reducts now determine smallest sets S such that $I_S(P,N) \neq \varnothing$.

Definition 3.3 A *reduct* for two concepts P and N is a minimal set of concept-names R to preserve existence of an interpolant, i.e. where $I_R(P, N) \neq \varnothing$ and $I_{R'}(P, N) = \varnothing$ for every $R' \subset R$.

The set $\{Doctor, Rich\}$ is not minimal, as $I_{\{Doctor\}}(C_{ex}, D_{ex}) \neq \varnothing$. On the other hand, $\{Doctor\}$ is a reduct. Also, each interpolant in $I_{\{Doctor\}}(C_{ex}, D_{ex})$ is optimal for C_{ex} and D_{ex}. This observation can be generalised:

Lemma 3.4 *Let R be a reduct for two concepts P and N. Every interpolant $I \in I_R(P, N)$ is optimal for P and N.*

This lemma, which is proved in Appendix A, allows to calculate reducts and interpolants separately, and is used in the algorithms in the next section.

4 Algorithms for Optimal Interpolation

Optimal interpolants will be constructed using Boolean minimisation of the concept-names needed to close an $\mathcal{ALC}$-tableau. This calculation can be split into three steps. First, we saturate a labelled tableau as described in Section 4.1. Secondly, we calculate reducts from tableau proofs using the algorithm of Fig. 2. Finally, optimal interpolants can be constructed from the tableau proofs and the reducts according to Fig. 3. Step 1 and 3 closely follow well-known procedures that can be found, e.g., in [2] (for the saturation of the tableau) and [9] (for the calculation of interpolants). New is the calculation of reducts in Section 4.2 and their application in Section 4.3 to ensure optimality of the calculated interpolants.

4.1 Saturating Labelled Tableaux

Concept subsumption $\models P \sqsubseteq N$ can be decided by a proof deriving a closed tableau starting from a tableau with one branch $\{(i : P)^p, (i : \neg N)^n\}$ (for an arbitrary individual i). The information whether a formula has its origin in P or N is needed to construct interpolants from the proof. Each formula stemming from P will be labelled with $(\cdot)^p$, each created from N with $(\cdot)^n$. A *formula* has the form $(i : C)^x$ where i is an individual, C a concept and $x \in \{p, n\}$ a label. The labelling mechanism follows [9]. A *branch* is a set of formulas and a *tableau* a set of branches. A formula can occur with different labels on the same branch. A branch is *closed* if it contains a clash, i.e. if there are formulas with contradictory atoms on the same individual. The notions of open/closed branches and tableaux are defined as usual and do not depend on the labels. Formulas are always assumed to be in *negation normal form*.

To calculate reducts and optimal interpolants for two concepts P and N we construct a proof from a tableau containing a branch $\{(i : P)^p, (i : \neg N)^n\}$ (for a new individual i) by applying the rules in Fig. 1 as long as possible. The rules are $\mathcal{ALC}$-tableau rules (adapted from those of [2]) and have to be read in the following way; suppose that there is a tableau $T = \{B, B_1, \ldots, B_n\}$

($\sqcap$): **if** $(i : C \sqcap D)^x \in B$, but not both $(i : C)^x \in B$ and $(i : D)^x \in B$
then $B' := B \cup \{(i : C)^x, (i : D)^x\}$.

($\sqcup$): **if** $(i : C \sqcup D)^x \in B$, but neither $(i : C)^x \in B$ nor $(i : D)^x \in B$.
then $B' := B \cup \{(i : C)^x\}$ and $B'' := B \cup \{(i : D)^x\}$.

($\exists$): **if** $(i : \exists r.C)^x \in B$, all other rules have been applied, and $\{(i : \forall r.C_1)^{x_1}, \ldots, (i : \forall r.C_n)^{x_n}\}$ are all universal formulas for i and r in B,
then $B' := \{(j : C)^x, (j : C_1)^{x_1}, \ldots, (j : C_n)^{x_n}\}$, where j is new in B.

Fig. 1. Tableau rules for saturating a labelled $\mathcal{ALC}$-tableau (similar to [2])

with $n + 1$ branches. Application of one of the rules on B yields the tableau $T' := \{B', B_1, \ldots, B_n\}$ for the ($\sqcap$) and ($\exists$) rule or $T'' := \{B', B'', B_1, \ldots, B_n\}$ in case the ($\sqcup$)-rule has been applied. Application of one of the rules is called *to expand* a tableau or a branch. If no more rule can be applied, branches and tableaux are *saturated*. Finally, a *proof* is a sequence of tableaux $T_1, \ldots, T_n$ where each T_{i+1} has been created by application of one of the rules in Fig. 1 on a branches $B \in T_i$ (for $i, i + 1 \in \{1, \ldots, n\}$), and where T_n is saturated.

4.2 Calculating Reducts

From a proof starting with $\{\{(i : P)^p, (i : \neg N)^n\}\}$ we find reducts by calculating a *maximal reduct-function*. To define such an interpolation-preserving propositional formula we need to introduce interpolants for branches and individuals. Let B be a branch, i an individual and $\{(i : C_1)^p, \ldots, (i : C_k)^p, (i : D_1)^n, \ldots, (i : D_l)^n\}$ the set of formulas for i in B. An *interpolant* for B and i is an interpolant for $C_1 \sqcap \ldots \sqcap C_k$ and $\neg D_1 \sqcup \ldots \sqcup \neg D_l$.

The set of interpolants for B and i will be denoted by $I(i, B)$, or $I_S(i, B)$ for the interpolants built from concept-names occurring in a particular set S only. Reduct-functions are then interpolation-preserving propositional formulas.

Definition 4.1 Let B be a branch in a proof, i an individual and φ a propositional formula built from conjunction, disjunction and propositional variables. Let, furthermore, $tr(v)$ denote the (unique) set of propositional variables true in a valuation v, called the *truth-set* of v. If $I(i, B) \neq \varnothing$, φ is a *reduct-function* for B and i if, and only if, $I_{tr(v)}(i, B) \neq \varnothing$ for any valuation $v(\varphi) = T$. Otherwise, i.e., if $I(i, B) = \varnothing$, $\bot$ is the only reduct-function.

The idea to calculate reducts is as follows: As reduct-functions determine the sets of concept-names preserving interpolation a smallest set of this kind is a reducts. If a reduct-function is maximal, i.e. implied by all reduct-function, its prime implicants determine precisely these most general, i.e. smallest sets of concept-names. [2] A maximal reduct-function is calculated from a tableau proof: all branches of the saturated tableau must close even with a reduced set of concept-names available for closure. Therefore at least one clash per

[2] A prime implicant of φ is the smallest conjunction of literals implying φ (see, e.g., [12]). The term *prime implicant* refers both to the conjunction and to the set of conjuncts.

if *rule* $= (\sqcap)$ has been applied on $(i : C \sqcap D)^{label}$ and B' is the new branch
$rf(i, B) := rf(i, B')$;
if *rule* $= (\sqcup)$ has been applied on $(i : C \sqcup D)^{label}$ and B' and B'' are new
$rf(i, B) := rf(i, B') \wedge rf(i, B'')$;
if *rule* $= (\exists)$ has been applied on $(i : \exists r.C)^{label}$, B' and j are new
$rf(i, B) := rf(i, B') \vee rf(j, B')$;
if no more rule can be applied (for arbitrary x and y)
$rf(i, B) := \top$ if there are formulas $(i : A)^x \in B, (i : \neg A)^y \in B$ s.t. $x = y$;
$rf(i, B) := \bigvee_{(i : A)^x \in B, (i : \neg A)^y \in B} A$ otherwise; i.e. if $(x \neq y)$.

Fig. 2. $rf(i, B)$: A maximal reduct-function for a branch B and individual i

branch needs to be retained and a maximal reduct-function is the disjunction of the concept-names in all clashes. For each branch in a tableau proof complex maximal reduct-functions are then constructed recursively according to Fig. 2.

Theorem 4.2 Let P and N be concepts and i an arbitrary individual-name. The prime implicants of $rf(i, \{(i : P)^p, (i : \neg N)^n\})$ as calculated by the algorithm described in Fig. 2 are the reducts for P and N.

Proof. The proof consists of three parts. First, we show that $rf(i, B)$ is a reduct-function for every branch B. The proof is by induction over the tableaux in a proof, where we construct interpolants for B and i from the truth-sets of the valuation making $rf(i, B)$ true. The rules for construction of interpolants correspond to those we will later give explicitly in Fig. 3. If B is saturated there are several cases: first, if there are contradicting atoms with positive or with negative labels only, the interpolant is $\bot$ or $\top$ respectively, and the reduct-function is $\top$. If there are only clashes on atoms which occur both positively and negatively labelled in B any of the literals occurring positively is an interpolant, and the maximal reduct function is the disjunction of all the corresponding concept-names. Finally, a branch without clashes on the individual i does not have an interpolant for B and i, and $\bot$ is the only reduct-function. If the branch B is not saturated, one of the rules of Fig. 2 must have been applied, and we can construct an interpolant for B and i from the interpolants of the newly created branches. If a disjunctive rule had been applied, two new branches B' and B'' have been created, and it can easily be checked that the disjunction of an arbitrary interpolant for B' and i with an arbitrary interpolant for B'' and i is an interpolant for B and i. The conjunctive case is even more simple, as any interpolant for the new branch B' and i is also an interpolant for B and i. The only slightly more complicated case is when an existential rule has been applied on a formula in B because we need to take into account that the interpolant for the new branch and the new individual might be $\bot$. In this case, however, we can show that $\bot$ is also an interpolant for B and i, which finishes the proof.

Next, we show that $rf(i, B)$ is maximal, again by induction over the tableaux in the proof. For each branch B we show that $\varphi \rightarrow rf(i, B)$ for each reduct-

function φ of i and B. Again, if B is saturated it is easy to show that *rf(i, B)* is maximal for each of the cases mentioned above. For a non-saturated B we again have branches B' (and possibly B''), and we can easily show that $\varphi\rightarrow$*rf(i, B)* whenever $\varphi\rightarrow$*rf(i, B′)* (and possibly $\varphi\rightarrow$*rf(i, B″)*).

Finally, we prove that the prime implicants of maximal reduct-functions for the branch $B = \{(i : P)^p, (i : \neg N)^n\}$ are the reducts for P and N. Here, we show that there is an interpolant for P and N, and that there is no interpolant for any subset of the reduct. The other direction, i.e. the fact that every reduct R is the prime-implicant of the maximal reduct-function of B, follows immediately from the minimality of the reducts. See Appendix A for the technical details of the proof. □

4.3 Calculating Optimal Interpolants

Given a tableau proof and a reduct R we construct optimal interpolants for each branch B and each individual i recursively according to the rules in Fig. 3. It is well-known how to calculate interpolants from a tableau proof [9,13]. If a rule had been applied on a formula $(i : C)^x$ in B for an arbitrary label x, and one or two new branches B' (and B'') have been created, the interpolant for B and i can be constructed from interpolants for B' (and B''). It can easily be checked that *oi*$(i, \{(i : P)^p, (i : \neg N)^n\}, R)$ is an interpolant for P and N. By Lemma 3.4 we now immediately know that if R is a concept-reduct for P and N, this interpolant is also optimal.

Theorem 4.3 Let R be a reduct for two concepts P and N. The concept *oi(i,* $\{(i : P)^p, (i : \neg N)^n\}$*, R)* as calculated by the algorithm defined in Fig. 3 is an optimal interpolant for P and N.

Proof. Lemma 3.4 states that the interpolants built from concept-names in reducts are the optimal interpolants. Therefore, it suffices to show that the rules in Fig. 3 produce interpolants. This proof follows Kracht's proof in [9], we simply construct interpolants recursively for each branch in the proof. Full details can be found in Appendix A. □

4.4 Complexity

Calculating interpolants for two $\mathcal{ALC}$ concepts P and N is in PSPACE as we can apply the algorithm described in Fig. 3 in a depth-first way on one branch (of maximally polynomial space) at a time. This gives a simple bottom-up procedure to calculate optimal interpolants in PSPACE: for all subsets of the concept-names occurring in P and N we check whether there are interpolants or not, starting with the smallest, systematically increasing the size. Each of these checks can be done using only polynomial space. As soon as we find the first interpolant, i.e. as soon as *oi(i, B, R)* is defined for some subset R, we know that R is a reduct, and the interpolant must be optimal. For a lower bound consider the subsumption relation $C \sqsubseteq \bot$ which has an interpolant if,

if *rule* = ($\sqcap$) has been applied on $(i : C \sqcap D)^{label}$ and B' is the new branch
  return *oi(*i, B', R*);*
if *rule* = ($\sqcup$) has been applied on $(i : C \sqcup D)^{label}$ and B' and B'' are new
  if $label = p$ **return** *oi(*i, B', R*)* $\sqcup$ *oi(*i, B'', R*);*
  else if $label = n$ **return** *oi(*i, B', R*)* $\sqcap$ *oi(*i, B'', R*);*
if *rule* = ($\exists$) has been applied on $(i : \exists r.C)^{label}$, B' and j are new
  if *oi(*i, B', R*)* exists
    if *oi(*j, B', R*)* exists
      if $label = p$ **if** *oi(*j, B', R*)*$= \bot$ **return** *oi(*i, B', R*);*
          else return *oi(*i, B', R*)* $\sqcup$ $\exists r.$*oi(*j, B', R*);*
      else if $label = n$ **if** *oi(*j, B', R*)*$=\top$ **return** $\top$
          else return *oi(*i, B', R*)* $\sqcup$ $\forall r.$*oi(*j, B', R*);*
    else return *oi(*i, B', R*);*
  else if *oi(*j, B', R*)* exists
      if $label = p$ **if** *oi(*j, B', R*)*$= \bot$ **return** $\bot$*;* **else return** $\exists r.$*oi(*j, B', R*);*
      else if $label = n$ **if** *oi(*j, B', R*)*$= \top$ **return** $\top$*;* **else return** $\forall r.$*oi(*j, B', R*);*
  else return *undefined;*
if no more rule can be applied. For all concept-names A in R.
  if there is a clash on a concept-name $A \in R$
      if there are formulas $(i : A)^n \in B$ and $(i : \neg A)^n \in B$ **return:** $\top$
      else if there are formulas $(i : A)^p \in B$ and $(i : \neg A)^p \in B$ **return:** $\bot$
      else return: $\bigsqcup_{(i:A)^p \in B, (i:\neg A)^n \in B, A \in R} A \sqcup \bigsqcup_{(i:A)^n \in B, (i:\neg A)^p \in B, A \in R} \neg A$
  else return: *undefined.*

Fig. 3. *oi(*i, B, R*)*: Optimal interpolants

and only if, C is unsatisfiable. This means that interpolation must be at least as hard as concept satisfiability in $\mathcal{ALC}$, which is well known to be PSPACE.

Although the problem of calculating optimal interpolants is in PSPACE the algorithm described above might be infeasible in practice. To be sure that we have calculated all optimal interpolants we might have to check all elements of the power-set of the set of concept-names, i.e., we might have to saturate an exponential number of tableaux. Instead, our approach expands a single tableau once, from which we calculate the reducts and read off the optimal interpolants. Computing reducts is the computational bottle-neck of our algorithm as we calculate prime implicants on formulas which can be exponential in the size of the concepts. Given that calculating prime implicants is NP-hard, we must ensure that the size of the reduct-function is as small as possible. Our current implementation comprises simple on-the-fly elimination of redundancies, but more evolved methods need to be investigated. Our algorithms have an exponential worst case complexity in the size of the concepts P and N. As the number of variables in the reduct-function is linear in the number of concept-names in P and N we can calculate the prime implicants in exponential time branch by branch (instead of constructing the full reduct-function first). This simple method requires exponential space as we have to keep maximally $e^{\frac{n}{e}}$ prime implicants (of size smaller than n) in memory, where n is the

number of concept-names in P and N and e the base of natural logarithm.

5 Evaluation

Optimal interpolants are not unique, and a certain leeway exists for choosing suitable interpolants according to a given application. The algorithm in Fig. 3 was developed with the learning application of Section 2 in mind and, for this reason, calculates very general interpolants. Whenever we have a choice we construct an interpolant in the most general way. This is the case when applying a ($\exists$)-rule as well as when calculating an interpolant for a saturated branch. Note that this choice might lead to rather complex concepts with a relatively big size. Our decision was to separate the two problems of optimal interpolation and rewriting of concepts with minimal size for conceptual clarity. If optimal interpolants are applied to explain a subsumption relation things might be different: first, the size of an explanation should be as small as possible and, secondly, we might also want to have different levels of generality, such as the most specific optimal interpolant. It is straightforward to adapt the algorithm of Fig. 3 to calculate smaller or more specific optimal interpolants and we plan to evaluate different strategies for both described applications in future research.

We implemented interpolation as part of the Wellington's KAT system [7] and applied it to learn a terminology about cardiac arrhythmias [8]. This implementation did not calculate optimal interpolants and over-generalises on previously unknown data. This motivated the current minimisation techniques. Although the problem of calculating interpolants for two concepts in $\mathcal{ALC}$ is in pspace the results were encouraging as interpolants could be calculated efficiently given the relatively simple structure of the application. Prototypical implementations of the algorithms for *optimal interpolants* confirm that prime-implicants are indeed difficult and that current method based on a reduction to integer programming is impractical. We are working on a more robust implementation. Given the high complexity, optimisation techniques to reduce the size of reduct-functions will have to be developed.

6 Conclusions

We introduce an algorithm to find interpolants with a minimal number of concept-names for two concepts in the description logic $\mathcal{ALC}$, with a rigid definition of the common vocabulary. The principal novelty is that we minimise the number of concept-names in order to find most simple interpolants. These optimal interpolants are used in applications as divers as explanation and learning of terminologies.

We are currently robustly implementing the algorithms, in order to evaluate learning and explanation by interpolation on real-life data. We also plan to extend the algorithms to more expressive languages. An open problem is

how to calculate interpolants of minimal size, as redundancy elimination will be crucial for optimal interpolants to be useful in newly learned terminologies or as illustrations.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[3] A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Explaining $\mathcal{ALC}$ subsumption. In *DL-99*, pages 37–40, 1999.

[4] W. Cohen and H. Hirsh. Learning the classic description logic: Theoretical and experimental results. In *KR-94*, pages 121–133, Bonn, Germany, 1994.

[5] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22:269–285, 1957.

[6] Minutes of the DL Implementation Group Workshop. `http://dl.kr.org/dig/minutes-012002.html`, visited on January 9, 2003.

[7] U. Endriss. Reasoning in description logics with WELLINGTON 1. 0. In *Proceedings of the Automated Reasoning Workshop 2000*, London, UK, 2000.

[8] H.A. Güvenir, B. Acar, G. Demiröz, and A. Cekin. A supervised machine learning algorithm for arrhythmia analysis. In *Computers in Cardiology*, volume 24, pages 433–436, 1997.

[9] M. Kracht. *Tools and Techniques in Modal Logic*. North Holland, 1999.

[10] R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *LNAI*. Springer, 2001.

[11] T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

[12] W.V. Quine. The problem of simplifying truth functions. *American Math. Monthly*, 59:521–531, 1952.

[13] W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12:403–423, 1983.

[14] S. Schlobach. *Knowledge Acquisition in Hybrid Knowledge Representation Systems*. PhD thesis, University of London, 2002.

[15] S. Schlobach. Optimal interpolation. Technical Report PP-2003-23, Universiteit van Amsterdam, ILLC, 2003. Beta Preprint Publication.

[16] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI*, 2003. Accepted for publication.

A Proofs

Proof of Lemma 3.4

Lemma 3.4 (reminder) Let R be a reduct for two concepts P and N. Every interpolant $I \in I_R(P, N)$ is optimal for P and N.

Proof. The proof is a straightforward consequence of the definitions. Let R be a reduct for P and N. This means, by definition, that $I_R(P, N) \neq \varnothing$ but $I_{R'}(P, N) = \varnothing$ for every $R' \subset R$. For each interpolant $I \in I_R(P, N)$ we show that there is no interpolant I' for P and N such that $\mathcal{N}(I') \subset \mathcal{N}(I)$. But this is obvious, as $\mathcal{N}(I) \subseteq R$ and therefore $\mathcal{N}(I') \subset R$. The interpolant I' would therefore be in $I_{\mathcal{N}(I')}(P, N)$, which contradicts the fact that $I_{R'}(P, N) = \varnothing$ for each proper subset R' of R. □

Proof of Theorem 4.2

Theorem 4.2 (reminder) Let P and N be concepts and i an arbitrary individual-name. The prime implicants of $rf(i, \{(i : P)^p, (i : \neg N)^n\})$ as calculated by the algorithm described in Fig. 2 are the reducts for P and N.

Proof. The proof consists of three parts. First, we show that $rf(i, B)$ is a reduct-function for every branch B in the proof. The proof is by induction over the tableaux in a proof by constructing interpolants for B and i from the truth-sets of the valuation making $rf(i, B)$ true. This is Lemma A.1 Next, in Lemma A.2 we show that $rf(a, B)$ is maximal, again by induction over the tableaux in the proof. Finally, we prove that the prime implicants of a maximal reduct-function for the branch $\{(i : P)^p, (i : \neg N)^n\}$ are the reducts for P and N. This Lemma A.3 finishes the proof of Theorem 4.2.

The first lemma shows that our algorithm returns reduct-functions.

Lemma A.1 Let B be a branch occurring in a proof $T_1, \ldots, T_n$, and let i be an arbitrary individual. The propositional formula $rf(i, B)$ as defined in Fig. 2 is a reduct-function for B and i.

Proof. To show that $rf(i, B)$ is a reduct-function for B and i we prove that for every valuation $v(rf(i, B)) = T$ there is an interpolant for B and i constructed from concept-names in $tr(v)$ only, i.e. that $I_{tr(v)}(i, B) \neq \varnothing$. The proof is by induction over the sequence of tableaux in a tableau proof:

Induction Hypothesis: For every tableau T in a tableau proof the propositional formula $rf(i, B)$ is a reduct-function for any branch $B \in T$ and individual i.

(i) The base-case is a saturated tableau, where no rule is applicable. Then, for every branch B, we have four cases:
   (a) $x = y = p$; There are formulas $(i : A)^p \in B$ and $(i : \neg A)^p \in B$. In this case $\bot$ is an interpolant for B and i, as
   - $A \sqcap \neg A \sqcap \ldots \sqsubseteq \bot \sqsubseteq C$ and

- $\mathcal{L}(\bot) \subseteq \mathcal{L}(A \sqcap \neg A \sqcap \ldots) \cap \mathcal{L}(C)$ for any concept C.

This implies that $I_{\varnothing}(i, B) \neq \varnothing$ and, therefore, that $I_{tr(v)}(i, B) \neq \varnothing$ for any valuation v. Thus, $rf(i, B) = \top$ is a reduct-function.

(b) $x = y = n$; There are formulas $(i : A)^n \in B$ and $(i : \neg A)^n \in B$. This case is dual to the previous one. In this case $\top$ is an object-related interpolant for B and i, as

- $C \sqsubseteq \top \sqsubseteq \neg A \sqcup A \sqcup \ldots$ and
- $\mathcal{L}(\top) \subseteq \mathcal{L}(C) \cap \mathcal{L}(\neg A \sqcup A \sqcup \ldots)$ for any concept C.

This implies that $I_{\varnothing}(i, B) \neq \varnothing$ and, therefore, that $I_{tr(v)}(i, B) \neq \varnothing$ for any valuation v. Thus, $rf(i, B) = \top$ is a reduct-function.

(c) $x \neq y$; There is no "internal" contradiction, but there are clashes involving positively and negatively labelled formulas. In this case $rf(i, B) = \bigvee_{(i:A)^x \in B, (i:\neg A)^y \in B} A$ is a reduct-function, as $v(rf(i, B)) = T$ implies for every valuation v that $tr(v)$ contains at least one concept-name A such that $(i : A)^x$ and $(i : \neg A)^y$, where $x \neq y$. This, however, means that either A (if $x = p$) or $\neg A$ (if $x = n$) is an interpolant for B and i. In both cases we know $I_{\{A\}}(i, B) \neq \varnothing$ for an arbitrary $A \in tr(v)$, and therefore also $I_{tr(v)}(i, B) \neq \varnothing$.

(d) Finally, if there is no clash, there is no interpolant for B and i, i.e., $I(i, B) = \varnothing$, and $\bot$ is a reduct-function by definition.

(ii) $\sqcup$-rule; A disjunctive rule was applied on a formula and two new branches B' and B'' have been created. We will prove that, $I_{tr(v)}(i, B) \neq \varnothing$ for every valuation such that $v(rf(i, B)) = T$. Let us construct such an interpolant.

By induction hypothesis $rf(i, B')$ and $rf(i, B'')$ are reduct-functions, i.e. $I_{tr(v')}(i, B') \neq \varnothing$ for any valuation $v'(rf(i, B')) = T$; and similarly for B''. As we know that $v(rf(i, B)) = T$ if, and only if, $v(rf(i, B')) = T$ and $v(rf(i, B'')) = T$ for any valuation v by the definition in Fig. 2 this implies $I_{tr(v)}(i, B') \neq \varnothing$ and $I_{tr(v)}(i, B'') \neq \varnothing$. Let us now choose two arbitrary interpolants $I' \in I_{tr(v)}(i, B')$ and $I'' \in I_{tr(v)}(i, B'')$ for each branch.

(a) Suppose that the $\sqcup$-rule was applied on a positively labelled formula and that $\{(i : C \sqcup D)^p, (i : C_1)^p, \ldots, (i : C_k)^p, (i : D_1)^n, \ldots, (i : D_l)^n\}$ is the set of formulas for i in the branch B. We will show that $I' \sqcup I''$ is an interpolant for B and i built from concept-names in $tr(v)$ only. We know that I' is an interpolant for $C \sqcap C_1 \sqcap \ldots \sqcap C_k$ and $\neg D_1 \sqcup \ldots \sqcup \neg D_l$, and that I'' is an interpolant for $D \sqcap C_1 \sqcap \ldots \sqcap C_k$ and $\neg D_1 \sqcup \ldots \sqcup \neg D_l$, i.e. that

- $C \sqcap C_1 \sqcap \ldots \sqcap C_k \sqsubseteq I' \sqsubseteq \neg D_1 \sqcup \ldots \sqcup \neg D_l$ and $D \sqcap C_1 \sqcap \ldots \sqcap C_k \sqsubseteq I'' \sqsubseteq \neg D_1 \sqcup \ldots \sqcup \neg D_l$. But then both $(C \sqcup D) \sqcap C_1 \sqcap \ldots \sqcap C_k \sqsubseteq I' \sqcup I''$ and $I' \sqcup I'' \sqsubseteq \neg D_1 \sqcup \ldots \sqcup \neg D_l$.
- Concerning the common vocabulary we know that $\mathcal{L}(I') \subseteq \mathcal{L}(C \sqcap C_1 \sqcap \ldots \sqcap C_k) \cap \mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l)$ and $\mathcal{L}(I'') \subseteq \mathcal{L}(D \sqcap C_1 \sqcap \ldots \sqcap C_k) \cap \mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l)$. As $\mathcal{L}(I) = \mathcal{L}(I' \sqcup I'') = \mathcal{L}(I') \cup \mathcal{L}(I'')$ it follows immediately that: $\mathcal{L}(I) \subseteq \mathcal{L}((C \sqcup D) \sqcap C_1 \sqcap \ldots \sqcap C_k) \cap$

$\mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l)$.

- Finally, $\mathcal{N}(I' \sqcup I'') \subseteq tr(v)$, as both $\mathcal{N}(I') \subseteq tr(v)$ and $\mathcal{N}(I'') \subseteq tr(v)$.

(b) Suppose that the $\sqcup$-rule was applied on a negatively labelled formula. This case is dual to the previous one, and $I' \sqcap I''$ is an interpolant for B and i built from concept-names in $tr(v)$ only.

(iii) $\sqcap$-rule; A conjunctive rule was applied on a formula and a new branch B' has been created. We will prove that $I_{tr(v)}(i, B) \neq \varnothing$ for every valuation such that $v(rf(i, B)) = T$. Let us construct such an interpolant.

By induction hypothesis $rf(i, B')$ is a reduct-functions, i.e. for any valuation $v'(rf(i, B')) = T$ implies that $I_{tr(v')}(i, B') \neq \varnothing$. As we know that $v(rf(i, B)) = T$ if, and only if, $v(rf(i, B')) = T$ for any valuation by the definition in Fig. 2 this implies $I_{tr(v)}(i, B') \neq \varnothing$. By definition of interpolation for a branch and an individual, every interpolant for B' is also an interpolant for B and vice versa. It follows immediately that $I_{tr(v)}(i, B) \neq \varnothing$.

(iv) $\exists$-rule; An existential rule was applied on a formula and a new branch B' has been created. We will prove that $I_{tr(v)}(i, B) \neq \varnothing$ for every valuation such that $v(rf(i, B)) = T$. Let us construct such an interpolant.

By induction hypothesis $rf(i, B')$ and $rf(j, B')$ are reduct-functions, i.e. $I_{tr(v')}(i, B') \neq \varnothing$ for any valuation $v'(rf(i, B')) = T$ and similar for j. We know that $v(rf(i, B)) = T$ if, and only if, $v(rf(i, B')) = T$ or $v(rf(j, B')) = T$ by the definition in Fig. 2. Therefore there must be either an $I' \in I_{v(tr)}(i, B')$ or $I'' \in I_{tr(v)}(j, B')$. It remains to show that either $I' \in I_{tr(v)}(i, B)$ or $I'' \in I_{tr(v)}(i, B)$ to finish the proof. There are now 2 main cases:

1.) Suppose a $\exists$-rule was applied on a positively labelled formula and

$$\{(i : \exists r.C)^p, (i : C_1)^p, \ldots, (i : C_k)^p, (i : \forall r.C_{k+1})^p, \ldots, (i : \forall r.C_m)^p, \\ (i : D_1)^n, \ldots, (i : D_l)^n, (i : \forall r.D_{l+1})^p, \ldots, (i : \forall r.D_o)^p\}$$

is the set of formulas for i in the branch B (where $C_1, \ldots, C_k$ and $D_1, \ldots, D_l$ are not universally quantified over r).

(a) Suppose I' is an interpolant for B' and i. As we add precisely one new formula for a new individual j to create the new branch B' it follows immediately that I' is also an interpolant for B and i.

(b) Suppose I'' is an interpolant for B' and j. This means that

- $C \sqcap C_{k+1} \sqcap \ldots \sqcap C_m \sqsubseteq I'' \sqsubseteq \neg D_{l+1} \sqcup \ldots \sqcup \neg D_o$, which in turns implies $\exists r.C \sqcap \forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m \sqsubseteq \exists r.I'' \sqsubseteq \exists r.\neg D_{l+1} \sqcup \ldots \sqcup \exists r.\neg D_o$. But this also implies that $\exists r.C \sqcap C_1 \sqcap \ldots \sqcap C_k \sqcap \forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m \sqsubseteq \exists r.I'' \sqsubseteq \neg D_1 \sqcup \ldots \sqcup D_l \sqcup \exists r.\neg D_{l+1} \sqcup \ldots \sqcup \exists r.\neg D_o$.
- We also have to show that:

$$\mathcal{L}(\exists r.I'') \subseteq \mathcal{L}(LHS) \cap \mathcal{L}(RHS) \tag{A.1}$$

where LHS is an abbreviation for the concept $\exists r.C \sqcap C_1 \sqcap \ldots \sqcap C_k \sqcap$

$\forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m$ on the *Left Hand Side* of the subsumption, RHS an abbreviation for the *Right Hand Side* $\neg D_1 \sqcup \ldots \sqcup \neg D_l \sqcup \exists r.\neg D_{l+1} \sqcup \ldots \sqcup \exists r.\neg D_o$. Remember that

$$\mathcal{L}(\exists r.I'') = \mathcal{L}(I'')^r \cup \{(\top, \_)^\epsilon, (\bot, \_)^\epsilon\} \tag{A.2}$$

according to Definition 3.1. We know by the definition of $\mathcal{L}$ that

$$\begin{aligned}\mathcal{L}(LHS) =& \mathcal{L}(\exists r.C \sqcap C_1 \sqcap \ldots \sqcap C_k \sqcap \forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m) \\ =& \mathcal{L}(C_1 \sqcap \ldots \sqcap C_k) \cup \mathcal{L}(\exists r.C) \\ & \cup \mathcal{L}(\forall r.C_{k+1}) \cup \ldots \cup \mathcal{L}(\forall r.C_m) \\ =& \mathcal{L}(C_1 \sqcap \ldots \sqcap C_k) \cup \mathcal{L}(C)^r \\ & \cup \mathcal{L}(C_{k+1})^r \cup \ldots \cup \mathcal{L}(C_m)^r \cup \{(\top, \_)^\epsilon, (\bot, \_)^\epsilon\}\end{aligned} \tag{A.3}$$

Because of the presence of $\exists r.C$ we know that $(\top, \_)^\epsilon$ and $(\bot, \_)^\epsilon$ are necessarily elements of $\mathcal{L}(LHS)$. This is not the case for $\mathcal{L}(RHS)$. Here we have to consider two cases.

– If $o > l + 1$ we also know that

$$\begin{aligned}\mathcal{L}(RHS) =& \mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l \sqcup \exists r.\neg D_{l+1} \sqcup \ldots \sqcup \exists r.\neg D_o) \\ =& \mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l) \cup \ \mathcal{L}(\exists r.\neg D_{l+1}) \cup \ldots \cup \mathcal{L}(\exists r.D_o) \\ =& \mathcal{L}(\neg D_1 \sqcup \ldots \sqcup \neg D_l) \\ & \cup \mathcal{L}(\neg D_{l+1})^r \cup \ldots \cup \mathcal{L}(\neg D_o)^r \cup \{(\top, \_)^\epsilon, (\bot, \_)^\epsilon\}\end{aligned} \tag{A.4}$$

But then our claim is true. By induction hypothesis $\mathcal{L}(I'') \subseteq \mathcal{L}(C \sqcap C_{k+1} \sqcap \ldots \sqcap C_m) \cap \mathcal{L}(\neg D_{l+1} \sqcup \ldots \sqcup \neg D_o)$, and $\mathcal{L}(I'')^r \subseteq \mathcal{L}(C \sqcap C_{k+1} \sqcap \ldots \sqcap C_m)^r \cap \mathcal{L}(\neg D_{l+1} \sqcup \ldots \sqcup \neg D_o)^r$ follows by definition of $\mathcal{L}(C)^r$ (where simply the role-name r is added to each sequence of role-names for each concept-name/polarity pair). Finally, this gives us, by definition of the conjunctive case of Definition 3.1,

$$\mathcal{L}(I'')^r \subseteq \mathcal{L}(LHS^r) \cap \mathcal{L}(RHS^r) \tag{A.5}$$

where LHS^r abbreviates $\mathcal{L}(C)^r \cup \mathcal{L}(C_{k+1})^r \cup \ldots \cup \mathcal{L}(C_m)^r$ and RHS^r abbreviates $\mathcal{L}(\neg D_{l+1})^r \cup \ldots \cup \mathcal{L}(\neg D_o)^r$.

The subset relation (A.1) follows from (A.5) and (A.2), as both $\mathcal{L}(LHS^r) \subseteq \mathcal{L}(LHS)$ and $\mathcal{L}(RHS^r) \subseteq \mathcal{L}(RHS)$ by (A.3) and (A.4).

– Otherwise (i.e. if $o = l + 1$), i.e. $\mathcal{L}(RHS') = \varnothing$ which implies that $(\top, \_)^\epsilon$ and $(\bot, \_)^\epsilon$ do not necessarily belong to the language of the right-hand side, i.e. $\mathcal{L}(RHS)$. This, however, was needed to prove relation (A.1) from (A.5). But it also means that the interpolant I'' for B' and j must be $\bot$, as by induction hypothesis

$$C \sqcap C_{k+1} \sqcap \ldots \sqcap C_m \sqsubseteq I'' \sqsubseteq \bot \tag{A.6}$$

(remember that the empty disjunction is equivalent to $\bot$). But in this special case $I = \bot$ can easily been shown to be an interpolant. The subsumption relation (A.6) and Definition 3.1 for $\mathcal{L}$ imply that

$$\begin{aligned}&\exists r.C \sqcap \forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m \sqsubseteq \exists r.I'' \sqsubseteq \bot \text{ and} \\ &\mathcal{L}(\bot) \subseteq \mathcal{L}(\exists r.C \sqcap \forall r.C_{k+1} \sqcap \ldots \sqcap \forall r.C_m) \cap \mathcal{L}(\bot).\end{aligned}$$

which finishes the proof.

2.) Suppose a $\exists$-rule was applied on a negatively labelled formula. This case is dual to the previous one, but this time we show that $\forall r.I''$ is an interpolant for B and i. Again, special care needs to be taken when there is no universally quantified concept, this time on the left-hand side of the subsumption. In this case $\top$ is an interpolant for B' and j, and therefore also for B and i.

(v) If a rule was applied on a formula $(j : C)^x$ for an arbitrary $j \neq i$, $C \in \mathcal{ALC}$ and label x the set of object-related interpolants for i does not change. This is even the case if two new branches are created, and it is sufficient to choose one.

This finishes the proof of Lemma A.1. □

Next we show that $rf(i, B)$ is maximal w.r.t. propositional consequence in the set of all reduct-functions, i.e. that $\varphi \rightarrow rf(i, B)$ is valid for any reduct-function φ for i and B.

Lemma A.2 Let B be a branch occurring in a tableau proof $T_1, \ldots, T_n$, and i an individual. The reduct-function $rf(i, B)$ as defined in Fig. 2 is a maximal reduct-function for B and i.

Proof. We have shown that $rf(i, B)$ is a reduct-function. The proof that it is maximal is by induction over the rules applied in a tableau proof. We will have to show that $\varphi \rightarrow rf(i, B)$ for every reduct function φ for B and i.

Induction Hypothesis: For every tableau T in a tableau proof the reduct-function $rf(i, B)$ is maximal for any branch $B \in T$ and individual i.

(i) In the base case, the tableau T is saturated and no rule can be applied. There are 4 cases:
   (a) $x = y$; There are formulas $(i : A)^x \in B$ and $(i : \neg A)^y \in B$ for $x = y$. As $rf(i, B) = \top$ the claim follows trivially.
   (b) $x \neq y$; There is no "internal" contradiction, but there are clashes involving positively and negatively labelled formulas. To close the branch, at least one $A \in tr(v)$ is needed in $tr(v)$ for each reduct-function. But then $\varphi \rightarrow \bigvee_{(i : A)^x \in B, (i : \neg A)^y \in B} A$.
   (c) Finally, if there is no clash, there is no interpolant, i.e. $I(i, B) = \varnothing$ and $\bot$ is the only reduct-function by definition.

(ii) Suppose a $\sqcup$-rule has been applied and two new branches B' and B'' were created. Every reduct function φ for B is also a reduct function for B' and B'', as $I_{tr(v)}(B, i) \neq \varnothing$ for any valuation $v(\varphi) = T$ implies that $I_{tr(v)}(B', i) \neq \varnothing$ and $I_{tr(v)}(B'', i) \neq \varnothing$. We will show in Lemma A.4 that the concept $oi(i, B, R)$ is undefined if, and only if, there is no interpolant I for i and B with $\mathcal{N}(I) \subseteq R$. This can only happen if there is an open branch on the saturated tableau derived from B. Assume, however, that there is an open branch for the saturated tableaux derived from either B' or B''. In this case the saturated tableau derived from B would have

an open branch, and there would be no interpolant for B and i with concept-names in R only. Take now $R = tr(v)$ and we get an immediate contradiction.

Therefore every reduct-function φ for B and i is also a reduct-function for B' and B'', and φ it must be maximal. This follows from the induction hypothesis as both $\varphi \rightarrow rf(i, B')$ and $\varphi \rightarrow rf(i, B'')$ are valid, which implies validity of $\varphi \rightarrow rf(i, B') \wedge rf(i, B'')$.

(iii) The conjunctive case follows immediately from the induction hypothesis as the set of reduct-functions for B and i is the same as the reduct functions for B' and i, because the interpolants for B and B' are identical for i.

(iv) Suppose a $\exists$-rule has been applied and created a new branch B' and a new individual b. Every reduct-function φ for B must be a reduct-function for B' and either i or j (or both), because, as before, the tableau derived from B must close on concept-names in $tr(v)$.

This finishes the proof as by induction hypothesis now either $\varphi \rightarrow rf(i, B')$ or $\varphi \rightarrow rf(j, B')$ are valid, which implies validity of $\varphi \rightarrow rf(i, B') \vee rf(j, B')$.

This finishes the proof of Lemma A.2. □

Lemma A.3 Let P and N be concepts and $P \sqsubseteq N$. Let φ be an maximal reduct-function for an arbitrary individual i and a branch $B = \{(i : P)^p, (i : \neg N)^n\}$. The set of prime implicants for φ is the set of reducts for P and N.

Proof. First we study two special cases, before moving to the general one.

- $\varphi = \bot$; by definition we have that $I(i, B) = \varnothing$, and therefore $I(P, N) = \varnothing$. But this implies $P \not\sqsubseteq N$, which is a contradiction to the assumption.
- $\varphi = \top$; ($\Rightarrow$) *Every prime implicant of φ is a reduct.* Every valuation makes φ true, i.e. also the valuation with an empty truth-set. This implies that $I_\varnothing(a, B) \neq \varnothing$. But $\varnothing$ is a subset of any set, and therefore of any truth-set of any satisfying valuation for φ. ($\Leftarrow$) *Every reduct for P and N is a prime implicant of φ*, i.e. the empty set $\varnothing$. That is trivial $\varnothing$ is already the smallest set to preserve existence of an interpolant for P and N.
- For all other cases we have to show that $R \in pi(\varphi)$ iff R is a reduct, i.e.:

(i) $I_R(P, N) \neq \varnothing$

(ii) $I_{R'}(P, N) = \varnothing$ for any set $R' \subset R$.

($\Rightarrow$) Suppose R is a prime implicant of φ. We will show that R fulfils Conditions i and ii. There is a valuation v' which makes $\bigwedge_{A \in R} A$ true, otherwise $R = \varnothing$ and therefore $\varphi = \top$, which we covered in the case above. As R is a prime implicant of φ the implication $\bigwedge_{A \in R} A \rightarrow \varphi$ is valid, and therefore $v(\bigwedge_{A \in R} A) = T$ implies $v(\varphi) = T$ for any valuation v. Therefore φ must also be true in v', which implies that $I_R(a, B) \neq \varnothing$ and therefore also $I_R(P, N) \neq \varnothing$ as φ is a reduct-function. This proves Condition i. Now take a subset $R' \subset R$. R is a prime implicant of φ, so that $\bigwedge_{A \in R'} A \rightarrow \varphi$ is not valid. Maximality of φ implies that $\bigwedge_{A \in R'} A$ is not a reduct-function. But

then, by definition, there must be a valuation v' such that $v'(\bigwedge_{A\in R'} A) = T$ and $I_{tr(v')}(i, B) = \varnothing$. But as $R' \subseteq tr(v')$ we must also have that $I_{R'}(a, B) = \varnothing$, which implies $I_{R'}(P, N) = \varnothing$.
($\Leftarrow$) Suppose R is a reduct. We show that R is a prime implicant of φ. Condition 1. implies that $I_R(a, B) \neq \varnothing$. From this follows that $\bigwedge_{A\in R} A$ is a reduct-function because $R \subseteq tr(v)$ for all valuations $v(\bigwedge_{A\in R} A) = T$. Maximality of φ implies validity of $\bigwedge_{A\in R} A \rightarrow \varphi$. But now we are done: assume that R is not prime, i.e. that there is a smaller set $R' \subset R$ where $\bigwedge_{A\in R'} \rightarrow \varphi$ is valid. This, however, implies that $I_{R'}(P, N) \neq \varnothing$, which contradicts Condition 2.

This finishes the proof of Lemma A.3. □

Theorem 4.2 now follows immediately from Lemmas A.1, A.2 and A.3. □

Proof of Theorem 4.3

Theorem 4.3 (reminder) Let R be a reduct for two concepts P and N. The concept $oi(i, \{(i : P)^p, (i : \neg N)^n\}, R)$ as calculated by the algorithm defined in Fig. 3 is an optimal interpolant for P and N.

Proof. As we have already noted that interpolants built from concept-names in reducts only, determine the set of optimal interpolants it suffices to show that the rules in Fig. 3 produce interpolants. We show, in Lemma A.4 that the concept $oi(i, B, R)$ is an interpolant for an arbitrary branch B in a proof and an individual i and that it is built from concept-names in R only. The proof is by induction over the tableaux in a proof, and we simply have to construct an interpolant for B from the branches B' (and B''). Note that we have some leeway in the construction of the interpolant, and that we usually chose the most general interpolant. [3]

Lemma A.4 Let B be a branch in a tableau proof, i be an individual name. If defined, the concept $oi(i, B, R)$ as calculated by the rules in Fig. 3 is an interpolant for B and i such that $\mathcal{N}(oi(i, B, R)) \subseteq R$. The concept $oi(i, B, R)$ is undefined if, and only if, there is no interpolant for B and i with $\mathcal{N}(oi(i, B, R)) \subseteq R$.

Proof. To prove Lemma A.1 we had to show that we could construct an interpolant for each branch consisting of the vocabulary only which occurred in the truth-set of a valuation making the reduct-function true. There, we have already seen how to construct the interpolants, and we use this information in the following proof. The proof of Lemma A.4 is by induction over the tableau proofs as before.

[3] This arbitrary choice is motivated by the application of optimal interpolants in learning of terminologies where we aim to learn most general TBox axioms.

Induction Hypothesis: For every tableau T in a proof and every branch $B \in T$ the concept *oi(*i, B, R*)* (if defined) as calculated by the algorithm in Fig. 3 is
(i) an interpolant for B and i
(ii) such that $\mathcal{N}(oi(i, B, R)) \subseteq R$.
If *oi(*i, B, R*)* is undefined there is no interpolant with $\mathcal{N}(oi(i, B, R)) \subseteq R$ for B and i.

(i) The base-case is a saturated tableau, where no rule is applicable. Then, for every branch B, we have four cases:
   (a) $x = y = p$; There are formulas $(i : A)^p \in B$ and $(i : \neg A)^p \in B$. In this case $\bot$ is an interpolant for B and i as shown in case 1.(a) of the proof of Lemma A.1. Condition (ii) follows trivially as $\mathcal{N}(\bot) = \varnothing$
   (b) $x = y = n$; There are formulas $(i : A)^n \in B$ and $(i : \neg A)^n \in B$. In this case $\top$ is an interpolant for B and i, as shown in case 1.(b) of the proof of Lemma A.1. Condition (ii) follows trivially as $\mathcal{N}(\bot) = \varnothing$
   (c) $x \neq y$; *oi(*i, B, R*)* is defined, but there is no "internal" contradiction, i.e. only clashes involving positively and negatively labelled formulas. Let P (and N) denote the conjunction (disjunction) of concepts occurring positively (negatively) labelled in B in formulas for the individual i. For every atom $A \in R$ where $(i : A)^x$ and $(i : \neg A)^y$ (with $x \neq y$) either A (if $x = p$) or $\neg A$ (if $x = n$) is an interpolant for B and i w.r.t. R, i.e. $P \sqsubseteq A \sqsubseteq N$ (if $x = p$) or $P \sqsubseteq \neg A \sqsubseteq N$ (if $x = y$). Then also $P \sqsubseteq \bigsqcup_{(i:A)^p \in B, (i:\neg A)^n \in B, A \in R} A \sqcup \bigsqcup_{(i:A)^n \in B, (i:\neg A)^p \in B, A \in R} A \sqsubseteq N$. As $\mathcal{L}(A) \subseteq \mathcal{L}(P) \cap \mathcal{L}(N)$ for all A is follows $\mathcal{L}(oi(i, B, R)) \subseteq \mathcal{L}(P) \cap \mathcal{L}(N)$ which implies that *oi(*i, B, R*)* is an interpolant for i and B. $\mathcal{N}(oi(i, B, R)) \subseteq R$ follows trivially.
   (d) Suppose *oi(*i, B, R*)* is undefined. This means that there is no clash in B on formulas containing i with concept-names in R. But then there can be no interpolant for i and B built from concept-names in R only.

(ii) $\sqcup$-rule; A disjunctive rule was applied on a formula and two new branches B' and B'' have been created. It was shown in case 2. of the proof of Lemma A.1 that the combination of two interpolants for B' and B' yield interpolant for B and i: a conjunctive combination if the rule was applied on a negatively labelled formula, a disjunctive combination otherwise. The language constraint $\mathcal{N}(oi(i, B, R)) \subseteq R$ follows trivially from the induction hypothesis.

(iii) $\sqcap$-rule; A conjunctive rule was applied on a formula and a new branches B' has been created. By definition every interpolant for B' and i is also an interpolant for B and i. Again, the language constraint follows trivially.

(iv) $\exists$-rule; An existential rule was applied on a formula and a new branches B' has been created.
   • Suppose the rules was applied on a positively labelled formula. It was

shown in case 4.(a) of the proof of Lemma A.1 that $\exists r.I'$ is an interpolant for B and i if I' is an interpolant for B' and j, and that I is an interpolant for B and i if it is an interpolant for B' and i. It is easy to check that $I \sqcup \exists r.I'$, I or $\exists r.I'$ are interpolants for B and i depending on the existence of the interpolants for B' and i and j. Again, the language constraint follows trivially by induction hypothesis.

- The same arguments hold if the rule was applied on a negatively labelled formula, this time with the interpolant $\forall r.I'$.

This finishes the proof of Lemma A.4. □

By definition of interpolation for branches and individuals it follows that $oi(i, \{(i : P)^p, (i : \neg N)^n\}, R)$ is an interpolant for P and N, and if R is a concept-reduct for P and N, this interpolant is optimal by Lemma 3.4.

This finishes the proof of Theorem 4.3. □