Uniform Interpolation of $\mathcal{ALC}$-Ontologies Using Fixpoints

Patrick Koopmann[*] and Renate A. Schmidt

The University of Manchester, UK
{koopmanp, schmidt}@cs.man.ac.uk

Abstract. We present a method to compute uniform interpolants with fixpoints for ontologies specified in the description logic $\mathcal{ALC}$. The aim of uniform interpolation is to reformulate an ontology such that it only uses a specified set of symbols, while preserving consequences that involve these symbols. It is known that in $\mathcal{ALC}$ uniform interpolants cannot always be finitely represented. Our method computes uniform interpolants for the target language $\mathcal{ALC}\mu$, which is $\mathcal{ALC}$ enriched with fixpoint operators, and always computes a finite representation. If the result does not involve fixpoint operators, it is the uniform interpolant in $\mathcal{ALC}$. The method focuses on eliminating concept symbols and combines resolution-based reasoning with an approach known from the area of second-order quantifier elimination to introduce fixpoint operators when needed. If fixpoint operators are not desired, it is possible to approximate the interpolant.

1 Introduction

Ontologies represent information about concepts and relations (roles) using description logics, fragments of first-order logic, to allow reasoning systems to derive implicit information automatically. There are situations where it is useful to restrict an ontology to a subset of the vocabulary without affecting the meaning of the remaining concepts. When reusing parts from a general ontology for a specific domain, this can be done by restricting the ontology to the concepts that are known and interesting in this domain. Instead of restricting an ontology to a more specific domain, another application is restricting the ontology to a set of higher level concepts to create a summary of the ontology. Another example is hiding confidential concepts, which is useful when an ontology is shared or published, but some information should be kept secret [9].

In uniform interpolation, the ontology is reformulated in such a way that only symbols from a specified set are used, while logical consequences over the remaining symbols are preserved [4]. This paper describes a method for uniform interpolation of ontologies represented in the description logic $\mathcal{ALC}$.

Uniform interpolation for $\mathcal{ALC}$ is not a new topic. In [18], a method based on tableaux reasoning was published. In [12] theoretical properties of uniform

[*] Patrick Koopmann is supported by an EPSRC EU Doctoral Training Award.

interpolation were presented, among them that uniform interpolants can in the worst case be of size triple exponential in the size of the original ontology.

A problem of uniform interpolation in $\mathcal{ALC}$ is that the interpolants cannot always be represented using a finite number of finite $\mathcal{ALC}$-axioms. We offer a solution to this problem by using $\mathcal{ALC}\mu$, which is $\mathcal{ALC}$ enriched with fixpoint operators [11], to represent interpolants. This way we show how to always compute a finitely represented interpolant. Since fixpoint operators are not common in the description logic community yet, we also describe ways of approximating the result, in case no adequate representation without fixpoints is possible.

Our method combines two approaches known from the context of second-order quantifier elimination [7]. First, we use a resolution-based calculus to eliminate symbols in a focused way. Resolution-based methods have been used for eliminating symbols in different logics, like first order logic [6] or modal logic [10], but these logics are not expressive enough if the result requires fixpoint operators. We use a clausal form based on structural transformation, where new concept symbols are introduced dynamically, in order to deal with this. Afterwards we use a variation of the generalised Ackermann's Lemma [14] to eliminate these introduced symbols and add fixpoint operators when necessary.

2 Preliminaries

Let N_c, N_r be two disjoint sets of *concept symbols* and *role symbols*. *Concepts* in $\mathcal{ALC}$ are of the following form:

$$\bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists r.C \mid \forall r.C,$$

where $A \in N_c$, $r \in N_r$ and C and D are arbitrary concepts. $\top$, $C \sqcap D$ and $\forall r.C$ are defined as abbreviations: $\top$ stands for $\neg\bot$, $C \sqcap D$ for $\neg(\neg C \sqcup \neg D)$ and $\forall r.C$ for $\neg\exists r.\neg C$.

A TBox is a set of *axioms* of the forms $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. $C \equiv D$ is a short-hand for the two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. Since we are only dealing with the TBox part of an ontology, we will use the terms 'ontology' and 'TBox' interchangeably.

The semantics of $\mathcal{ALC}$ is defined as follows. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where the *domain* $\Delta^{\mathcal{I}}$ is a nonempty set and the *interpretation function* $\cdot^{\mathcal{I}}$ assigns to each concept symbol $A \in N_c$ a subset of $\Delta^{\mathcal{I}}$ and to each role symbol $r \in N_r$ a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concepts as follows:

$$\bot^{\mathcal{I}} := \emptyset \qquad (\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}.$$

$C \sqsubseteq D$ is *true* in an interpretation $\mathcal{I}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is model of a TBox $\mathcal{T}$ if all axioms in $\mathcal{T}$ are true in $\mathcal{I}$. A TBox $\mathcal{T}$ is *satisfiable* if there exists a model for $\mathcal{T}$, otherwise it is *unsatisfiable*. $\mathcal{T} \models C \sqsubseteq D$ holds iff in every model of $\mathcal{T}$ we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are *equi-satisfiable* if every model of $\mathcal{T}_1$ can be extended to a model of $\mathcal{T}_2$, and vice versa.

In order to define $\mathcal{ALC}\mu$, we extend the language with a set N_v of *concept variables.* $\mathcal{ALC}\mu$ extends $\mathcal{ALC}$ with concepts of the form $\mu X.C$ and $\nu X.C$, where $X \in N_v$, and C is a concept in which X occurs as a concept symbol only positively (under an even number of negations). $\mu X.C$ is the *least fixpoint* of C on X and $\nu X.C$ the *greatest fixpoint.*

A concept variable X is *bound* if it occurs in the scope C of a fixpoint expression $\mu X.C$ or $\nu X.C$. Otherwise it is *free.* A concept is *closed* if it does not contain any free variables. Axioms in $\mathcal{ALC}\mu$ are of the form $C \sqsubseteq D$ and $C \equiv D$, where C and D are closed concepts.

Following [3], we define the semantics of fixpoint expressions. Let $\mathcal{V}$ be an *assignment function* that maps concept variables to subsets of $\Delta^{\mathcal{I}}$. $\mathcal{V}[X \mapsto W]$ denotes $\mathcal{V}$ modified by setting $\mathcal{V}(X) = W$. $C^{\mathcal{I},\mathcal{V}}$ is the interpretation of C taking into account this assignment, and when $\mathcal{V}$ is defined for all variables in C, $C^{\mathcal{I},\mathcal{V}} = C^{\mathcal{I}}$. The semantics of fixpoint concepts is defined as follows:

$$(\mu X.C)^{\mathcal{I},\mathcal{V}} := \bigcap\{W \subseteq \Delta^{\mathcal{I}} \mid C^{\mathcal{I},\mathcal{V}[X\mapsto W]} \subseteq W\}$$
$$(\nu X.C)^{\mathcal{I},\mathcal{V}} := \bigcup\{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I},\mathcal{V}[X\mapsto W]}\}.$$

3 Overview of the Method

We are interested in computing *uniform interpolants* of TBoxes. Let *sig*(C) denote the set of concept symbols occurring in the concept C and *sig*$(\mathcal{T})$ the set of concept symbols occurring in the TBox $\mathcal{T}$.

Definition 1. *Given a TBox $\mathcal{T}$ and a set Σ of concept symbols, the TBox $\mathcal{T}'$ is a* uniform interpolant *of $\mathcal{T}$ over Σ iff*

1. *sig*$(\mathcal{T}') \subseteq \Sigma$, *and*
2. *for every $C \sqsubseteq D$ with sig$(C \sqcap D) \subseteq \Sigma$: $\mathcal{T}' \models C \sqsubseteq D$ iff $\mathcal{T} \models C \sqsubseteq D$.*

Observe that from this definition follows that uniform interpolants of a TBox over a given set of concept symbols are unique modulo logical equivalence. Figure 1 gives an outline of our method for computing uniform interpolants. In Phase 1, the ontology is transformed into a set of clauses. In Phase 2, we process each concept symbol A that occurs in the TBox but not in the set Σ one after another. We saturate the set of clauses with respect to A using a set of rules and eliminate clauses containing A. This is described in more detail in Section 4. Both phases may introduce new symbols, which are eliminated Phase 3, which is described in more detail in Section 5. This phase may involve the use of fixpoint operators, if these symbols are cyclic. After this phase, the uniform interpolant is already computed, but we add a fourth phase that applies simplifications and converts resulting axioms to proper subsumption relations.

The order in which symbols are processed in Phase 2 and 3 is not crucial, since we can prove that the result is always the correct uniform interpolant. Different

```
1. Compute the clausal representation $N := clauses(\mathcal{T})$ of $\mathcal{T}$.
2. For every $A \in sig(\mathcal{T}) \setminus \Sigma$:
   – eliminate $A$ by setting $N := ELIM_{Res}(N, A)$. $ELIM_{Res}$ uses a resolution
     based procedure described in Section 4. This step introduces definer symbols.
3. Set $\mathcal{T} = N$. For every definer $D$ in the resulting clause set:
   – eliminate $D$ by setting $\mathcal{T} := ELIM_{Ack}(\mathcal{T}, D)$. $ELIM_{Ack}$ uses Ackermann's
     Lemma and may introduce fixpoints. This is described in Section 5.
4. Apply further simplifications to $\mathcal{T}$ if possible and return the resulting ontology.
```

Fig. 1. The complete method for computing uniform interpolants.

orders of symbol elimination may lead to different syntactic representations, but these are logically equivalent. The following theorem states the correctness of our method and is proven in Section 7.

Theorem 1. *For any $\mathcal{ALC}$ TBox $\mathcal{T}$ and any signature Σ, our method terminates and returns a finite representation of the uniform interpolant of $\mathcal{T}$ over Σ in the description logic $\mathcal{ALC}\mu$. If the result does not make use of the greatest fixpoint operator, the result is the uniform interpolant of $\mathcal{T}$ over Σ in $\mathcal{ALC}$.*

4 Resolution-Based Symbol Elimination

In this section we describe the first two phases of our method in more detail, that is, transformation to clausal form and elimination of symbols. The latter is based on a new resolution calculus for deciding satisfiability in $\mathcal{ALC}$, which we describe as well.

Let $N_D \subseteq N_c$ be a set of designated concept symbols called *definers*, which do not occur in the input ontology.

Definition 2. *An* $\mathcal{ALC}$-literal *is a concept description of the form A, $\neg A$, $\forall r.D$ or $\exists r.D$, where A is a concept symbol, r is a role symbol and D is a definer.*

A TBox is in $\mathcal{ALC}$-conjunctive normal form *if every axiom is of the form*

$$\top \sqsubseteq L_1 \sqcup ... \sqcup L_n,$$

where each L_i is an $\mathcal{ALC}$-literal. The right part of such a subsumption is called $\mathcal{ALC}$-clause. *In the following we assume $\mathcal{ALC}$-clauses are represented as sets of literals (this means no clause contains the same literal more than once). The empty clause is denoted by $\bot$ and represents a contradiction.*

Every $\mathcal{ALC}$ TBox can be transformed into an equi-satisfiable TBox in $\mathcal{ALC}$-conjunctive normal form using structural transformation. This can be achieved by first transforming the input TBox into negation normal form, incrementally replacing every concept C that occurs immediately below a role restriction by a

Resolution:

$$\frac{C_1 \sqcup A \qquad C_2 \sqcup \neg A}{C_1 \sqcup C_2}$$

provided $C_1 \cup C_2$ does not contain more than one negative definer-literal.

Role Propagation:

$$\frac{C_1 \sqcup \forall r.D_1 \qquad C_2 \sqcup Qr.D_2}{C_1 \sqcup C_2 \sqcup Qr.D_3}$$

where $Q \in \{\exists, \forall\}$ and D_3 is a (possibly new) definer representing $D_1 \sqcap D_2$, provided $C_1 \cup C_2$ does not contain more than one negative definer-literal.

Existential Role Restriction Elimination:

$$\frac{C \sqcup \exists R.D \qquad \neg D}{C}$$

Fig. 2. Rules of decision procedure $RES_{\mathcal{ALC}}$

definer D and adding the axiom $D \sqsubseteq C$ for each such subconcept. The resulting TBox does not contain any nested role restrictions and can be brought into $\mathcal{ALC}$-conjunctive normal form by applying standard CNF-transformation techniques. It is crucial for our method of computing uniform interpolants that the structural transformation is performed in this way.

For a TBox $\mathcal{T}$, let *clauses*($\mathcal{T}$) refer to the set of clauses generated in this way. The set *clauses*($\mathcal{T}$) is produced by Phase 1.

Example 1. Consider the following TBox $\mathcal{T}$:

$$A \sqsubseteq B \sqcup C \qquad B \sqsubseteq \exists r.B \qquad C \sqsubseteq \forall r.\neg B$$

The obtained clause set *clauses*($\mathcal{T}$) is the following:

1. $\neg A \sqcup B \sqcup C$
2. $\neg B \sqcup \exists r.D_1$
3. $\neg D_1 \sqcup B$
4. $\neg C \sqcup \forall r.D_2$
5. $\neg D_2 \sqcup \neg B$

where D_1 and D_2 are definers introduced during the structural transformation.

The resolution method used to eliminate symbols in Phase 2 is based on a new resolution-based decision procedure, $RES_{\mathcal{ALC}}$, which we describe next. $RES_{\mathcal{ALC}}$ uses the rules shown in Figure 2. The *resolution rule* is a variation of the classical resolution rule for propositional logic. Its side condition ensures that every derived clause contains at most one negative definer literal, a property needed for the successful elimination of the introduced definer symbols in Phase 3. Another motivation for this side condition is that clauses with different negative definer literals represent concepts occurring under different role restrictions. A combination of these only makes sense if the contexts of these role restrictions get combined as well. This is performed by the *role propagation rule*: it propagates the conceptual information under a universal role restriction into concepts occurring under other role restrictions, and creates a clause that represents the combined contexts of the definers. The role propagation rule is based on the logical entailment $\models ((A \sqcup B) \sqcap (C \sqcup D)) \sqsubseteq (A \sqcup C \sqcup (B \sqcap D))$.

Since the normal form has to be preserved, the role propagation rule may require the introduction of a new definer symbol D_3 representing the conjunction of the definers D_1 and D_2 occurring in the premises. This is done by adding new clauses $\neg D_3 \sqcup D_1$ and $\neg D_3 \sqcup D_2$ to the clause set. We refer to this as *combining D_1 and D_2 into a new definer D_3*. Observe that the resolution rule also applies to definer-literals. This way for each pair of clauses $\neg D_1 \sqcup C_1$ and $\neg D_2 \sqcup C_2$ we derive the clauses $\neg D_3 \sqcup C_1$ and $\neg D_3 \sqcup C_2$, for which the side conditions of the rules are satisfied.

In order to avoid the infinite introduction of new definers, we keep track of introduced definers and reuse them when possible. Let N_D* denote the definer symbols that were introduced by the initial normal form transformation in Phase 1, which we call *base definers*. The mapping *conj* : $N_D \mapsto 2^{N_D*}$ maps each definer to the set of base definers it represents. If a definer representing $D_1 \sqcap D_2$ is needed, we check whether the mapping already maps a definer to $conj(D_1) \cup conj(D_2)$. If it does, we reuse it; if not, we add a new definer together with the required axioms. This way the number of introduced symbols is bounded by $2^{|N_D*|}$.

It is not hard to see that the rules in Figure 2 are sound. This means saturating a set of clauses always produces an equi-satisfiable set of clauses. The existential role restriction elimination rule is sound, because the clause $\neg D$ represents the axiom $D \sqsubseteq \bot$. Since the introduction of definers preserves equi-satisfiability, we can state soundness of the calculus:

Lemma 1. *The calculus $RES_{\mathcal{ALC}}$ is sound, that is, for any TBox $\mathcal{T}$, the saturation of clauses($\mathcal{T}$) using the rules of $RES_{\mathcal{ALC}}$ is equi-satisfiable with $\mathcal{T}$, and if the empty clause can be derived, $\mathcal{T}$ is unsatisfiable.*

We can also prove refutational completeness and termination. The proof is given Section 6.

Theorem 2. *$RES_{\mathcal{ALC}}$ is sound and refutationally complete, and provides a decision procedure for TBox satisfiability in $\mathcal{ALC}$.*

This result is used in Section 7 to prove the correctness of our method to compute uniform interpolants.

The rules of $RES_{\mathcal{ALC}}$ are used in Phase 2 for saturating and eliminating symbols. In particular, in order to eliminate a concept symbol A from a set N of $\mathcal{ALC}$-clauses, we restrict the rules to be applied only on the literals $\neg A$, A, $\neg D$, D, $\forall r.D$ and $\exists r.D$, where A is the symbol we want to eliminate and D is a definer connected to A. A definer D is *connected to a concept symbol A* if D either co-occurs with A in a clause or if D co-occurs in a clause with another definer D' that is connected to A. After N is saturated using these restricted rules, we remove all clauses containing A and all clauses of the form $\neg D \sqcup D'$, where D and D' is any definer, since they are not required anymore. We call this method $\mathit{ELIM}_{\mathit{Res}}$ and denote the resulting set of clauses by $\mathit{ELIM}_{\mathit{Res}}(N, A)$.

Theorem 3. *Given the clausal representation N of a TBox $\mathcal{T}$ and a concept symbol A, $\mathit{ELIM}_{\mathit{Res}}(N, A)$ is computed in finitely bounded time, does not contain A and preserves all consequences over $\Sigma = \mathit{sig}(\mathcal{T}) \setminus \{A\}$.*

It is worth mentioning that both $\mathit{RES}_{\mathcal{ALC}}$ and $\mathit{ELIM}_{\mathit{Res}}$ can make use of standard redundancy elimination techniques used in resolution-based theorem proving, including tautology and subsumption deletion, which we omit here for space reasons.

Example 2. We demonstrate the application of $\mathit{RES}_{\mathcal{ALC}}$ on the clause set generated in the last example. Suppose we want to compute the uniform interpolant over $\Sigma = \{A, C\}$, which means B is the only concept symbol we have to eliminate. Resolution on the B-literals in clauses 3 and 5 would produce a clause with two different negative definer-literals, thus violating the side condition of the resolution rule (see Figure 2). But we can combine the definers D_1 and D_2 by applying the role propagation rule. Note that both are connected to B, and that the role propagation rule is applicable to clauses 2 and 4. This leads to the introduction of the definer D_3 representing $D_1 \sqcap D_2$, and the clauses capturing $D_3 \sqsubseteq D_1 \sqcap D_2$.

6.	$\neg B \sqcup \neg C \sqcup \exists r.D_3$	(*role prop. between 2 and 4*)
7.	$\neg D_3 \sqcup D_1$	$(D_3 \sqsubseteq D_1)$
8.	$\neg D_3 \sqcup D_2$	$(D_3 \sqsubseteq D_2)$

Observe that an additional application of the role propagation rule on the same clauses does not result in new clauses, since a definer representing $D_1 \sqcap D_2$ has already been introduced. The new clauses 7 and 8 can now be resolved on the positive definer literals.

9.	$\neg D_3 \sqcup B$	(*resolution between 3 and 7*)
10.	$\neg D_3 \sqcup \neg B$	(*resolution between 5 and 8*)

Now we have two clauses that allow resolving on B, resulting in a clause that makes the existential role restriction elimination rule applicable:

11.	$\neg D_3$	(*resolution between 9 and 10*)
12.	$\neg B \sqcup \neg C$	(*exist. elim. between 6 and 11*)

The last clause expresses the disjointness of the concepts B and C, which is a consequence of the last two axioms of the sample TBox. Further applications of the resolution rule are possible, which we omit for space reasons. In this example

5 Eliminating Definers Using Ackermann's Lemma

In Phase 3, the definers that have been introduced in Phase 2 are eliminated. This may involve the introduction of fixpoint operators. We also describe how to approximate the uniform interpolant in $\mathcal{ALC}$.

The main idea of this phase is captured in the following theorem:

Theorem 4. *Let $\mathcal{T}$ be a TBox which contains an axiom of the form $A \sqsubseteq C$, where A is a concept symbol that occurs only positively in the rest of $\mathcal{T}$.*

(i) If C does not contain A, the uniform interpolant of $\mathcal{T}$ over $sig(\mathcal{T}) \setminus \{A\}$ is obtained by removing that axiom and replacing every other occurrence of A in the rest of the ontology by C.

(ii) If C contains A positively, the interpolant is obtained by removing that axiom and replacing every occurrence of A with $\nu X.C'$, where C' is acquired from C by replacing every A with the fresh concept variable X.

This theorem is a translation of Ackermann's Lemma, which was first published in [1] and generalised for the fixpoint case in [14], to description logic syntax. Ackermann's Lemma and its generalisation have been used in the context of second-order quantifier elimination to eliminate existentially quantified predicate variables in second-order logic expressions [14, 7].

The underlying idea of the theorem is that if there is a definition of A in the ontology, we can use this definition to replace all occurrences of A in order to eliminate A. If the definition is cyclic, we have to use a fixpoint operator.

We use this theorem in Phase 3 to eliminate the introduced definer symbols. The method to compute $ELIM_{Ack}(\mathcal{T}, D)$ consists of the following steps:

1. Group all axioms of the form $\top \sqsubseteq \neg D \sqcup C_i$ into a single axiom of the form $D \sqsubseteq C_D$, where $C_D = \bigsqcap_i C_i$. If there is no such clause, set $C_D = \top$.
2. Remove the axiom $D \sqsubseteq C_D$ from $\mathcal{T}$.
3. If D does not occur in C_D, replace every occurrence of D in $\mathcal{T}$ with C_D.
4. If D occurs in C_D, replace every occurrence of D in $\mathcal{T}$ with $\nu X.C'_D$, where C'_D is acquired from C_D by replacing D with X, where X is a fresh concept variable not used in $\mathcal{T}$.

If the output of the algorithm contains fixpoints there are two ways in which we can approximate the result in $\mathcal{ALC}$: signature approximation and semantic approximation. In *signature approximation*, we return a finite TBox equi-satisfiable with the uniform interpolant which approximates the signature Σ and therefore may contain additional concept symbols. This is done by not eliminating definers which would lead to the use of fixpoint operators.

In contrast, using *semantic approximation*, we return a result that is completely in the specified signature Σ, but approximates the interpolant semantically. For this, we omit Step 3 and apply Step 2 above incrementally for a specified number of times even if D occurs in C_D, and replace it afterwards by $\top$. This way the semantics of the greatest fixpoint operator is approximated in the result. This solution is similar to the one offered in [18].

Example 3. We continue on the last example. The result of $\mathit{ELIM}_{\mathit{Res}}(N, B)$ is equivalent to the following ontology.[1]

1. $\top \sqsubseteq \neg A \sqcup C \sqcup \exists r.D_1$
2. $\top \sqsubseteq \neg C \sqcup \forall r.D_2$
3. $D_1 \sqsubseteq \neg C \sqcap \exists r.D_1$
4. $D_2 \sqsubseteq \neg A \sqcup C$
5. $D_3 \sqsubseteq \bot$

Axiom 5 can be ignored since D_3 does not occur in the rest of the ontology. D_2 can be eliminated by replacing it with $\neg A \sqcup C$. The elimination of D_1 leads to the introduction of a fixpoint operator. After applying simplifications (Phase 4), we obtain the following ontology, which is the uniform interpolant of our sample TBox for $\Sigma = \{A, C\}$:

6. $A \sqsubseteq C \sqcup \exists r.\nu X.(\neg C \sqcap \exists r.X)$
7. $C \sqsubseteq \forall r.(\neg A \sqcup C)$

We cannot express this uniform interpolant in a finite way in $\mathcal{ALC}$, but we can approximate it signature-wise and semantically. The signature approximation is acquired by not eliminating D_1 and including Axiom 3 in the result. For the semantic approximation, we would replace D_1 n times by $C \sqcap \exists r.D_1$, and then replace it by $\top$. For $n = 2$, Axiom 6 would be approximated as follows:

$$6'.\ A \sqsubseteq C \sqcup \exists r.(\neg C \sqcap \exists r.(\neg C \sqcap \exists r.\top)).$$

6 Correctness of the Decision Procedure

In this section we prove termination and refutational completeness of $\mathit{RES}_{\mathcal{ALC}}$. This result is needed in the next section to prove the correctness of $\mathit{ELIM}_{\mathit{Res}}$.

Lemma 2. *$\mathit{RES}_{\mathcal{ALC}}$ always terminates and produces at most $2^{2n_c+(2+2n_r)\cdot 2^{n_d}}$ clauses, where n_c is the number of concept symbols in the input ontology, n_r the number of role symbols and n_d the number of base definers introduced by the normal form transformation.*

Proof. Because of how we keep track of newly introduced definers, we have maximally 2^{n_d} many definers in the result. Each definer D can occur in the forms D, $\neg D$, $\exists r.D$ and $\forall r.D$, and each concept symbol A can occur in a positive or a negative literal, which means there are $2n_c + (2 + 2n_r) \cdot 2^{n_d}$ many possible literals. Every literal can only occur once in a clause (clauses are represented as sets), which gives us the worst case upper bound of $2^{2n_c+(2+2n_r)\cdot 2^{n_d}}$. □

In order to prove completeness of $\mathit{RES}_{\mathcal{ALC}}$, we use a candidate model construction approach similar to the one used to prove refutational completeness for ordered resolution [2] and refutational completeness for consequence-driven reasoners for description logics [16]. We show that for each set of clauses saturated using the rules of our calculus and not containing the empty clause, we can construct a candidate model which is actually a model for the set.

[1] To simplify the example, we left out redundant and tautological clauses, which would otherwise be removed in Phase 4 by the described method.

The construction is done in the following way. For each satisfiable definer concept D, we create a set I^D of literals that have to be satisfied by a domain element in order to satisfy the definer. A special definer ϵ is used to represent concepts that do not occur under a role restriction. We then create a domain element x_D for each definer D and construct an interpretation in such a way that every atomic concept and every existential restriction in each I^D is satisfied. We show that the resulting interpretation is indeed a model for our saturated set of clauses.

Let N_s denote a set of clauses saturated using the rules of $RES_{\mathcal{ALC}}$. The set $\mathbf{D}$ consists of all definers used in N_s and the special symbol ϵ. N_s is partitioned into a set of *definition sets*: the function $d : \mathbf{D} \longrightarrow 2^{N_s}$ maps each definer $D \in \mathbf{D}$ to the subset of clauses in N_s which have $\neg D$ as a literal, and ϵ to all remaining clauses. Because of the side conditions of the rules (every derived clause can have at most one negative definer-literal), such a partitioning is always possible. $d(D)$ contains all clauses that make up the *definition* of D, in the sense that they can be represented in an axiom of the form $D \sqsubseteq ...$, hence we use the terminology *definition set* for $d(D)$. $d(\epsilon)$ contains all the remaining clauses, which are not related to the definition of any definer. Let $d^e(D) = d(D) \cup d(\epsilon)$ be the definition set extended with these clauses. If a domain element satisfies D, it also has to satisfy all clauses in $d^e(D)$, and it suffices to check the clauses in $d^e(D)$ to check whether an instance satisfies D or not.

We define a partial ordering $\sqsubseteq_D$ on definers in the following way: $D_1 \sqsubseteq_D D_2$ iff $conj(D_2) \subseteq conj(D_1)$ (see Section 4 for the definition of $conj$). This ordering represents the subsumption hierarchy between the respective conjunctions of base-definers, because $\models \bigsqcap conj(D_1) \sqsubseteq \bigsqcap conj(D_2)$ if $conj(D_2) \subseteq conj(D_1)$.

We define an ordering $\prec_L$ on literals that satisfies the following constraints:

- $D \prec_L \neg D \prec_L A \prec_L \neg A \prec_L \exists r.D' \prec_L \forall r.D''$ for all atomic concepts A that are not definers, for all roles r and for all definers D, D', D''.
- If $D_1 \sqsubseteq_D D_2$, then $D_1 \prec_L D_2$, $\exists r.D_1 \prec_L \exists r.D_2$ and $\forall r.D_1 \prec_L \forall r.D_2$ for all roles r. (From this follows that if D represents $D_1 \sqcap D_2$, then $\exists r.D \prec_L \exists r.D_1$ and $\forall r.D \prec_L \forall r.D_2$.)

It can be shown that an ordering with these constraints always exists. $\prec_L$ is extended to an ordering $\prec_C$ between clauses using the multiset-extension $(\prec_L)_{mul}$ of $\prec_L$. Using $\prec_C$, the clauses in each $d^e(D)$ are enumerated: C_i^D denotes the ith clause in $d^e(D)$ according to $\prec_C$, starting from the smallest clause.

Following this enumeration, we define a set I^D of positive literals for each element $D \in \mathbf{D}$ (including ϵ), such that if a domain element x satisfies every literal in I^D, it also satisfies $d^e(D)$. For a set of positive literals I, we say *I satisfies a literal L, taking into account the subsumption hierarchy on $\boldsymbol{D}$*, written $I \models_{\mathbf{D}} L$, iff (i) L is a positive literal of the form A and $A \in I$, (ii) L is a negative literal of the form $\neg A$ and $A \notin I$, (iii) L is of the form $\exists r.D$ and there is a $\exists r.D' \in I$ with $D' \sqsubseteq_D D$ or (iv) L is of the form $\forall r.D$ and for every literal of the form $\exists r.D' \in I$ we have $D' \sqsubseteq_D D$. We say *I satisfies a clause C*, written $I \models_{\mathbf{D}} C$, if there is a literal $L \in C$ such that $I \models_{\mathbf{D}} L$.

We define I^D formally in five steps:

1. If $\neg D \in d(D)$, set $I^D = \emptyset$. Otherwise, let
2. $I_0^D = \{D\}$ if $D \neq \epsilon$ and $I_0^D = \emptyset$ if $D = \epsilon$.
3. $I_i^D = I_{i-1}^D \cup \{L\}$, if $I_{i-1}^D \not\models_{\mathbf{D}} C_{i-1}^D$ and the maximal literal L of C_{i-1}^D is a positive literal of the form A or $\exists r.D'$, and
4. $I_i^D = I_{i-1}^D$ otherwise.
5. $I^D = I_n^D$, where n is the number of clauses in $d^e(D)$.

Lemma 3. *If I^D is nonempty, then $I^D \models_{\boldsymbol{D}} C_i^D$ for all clauses C_i^D in $d^e(D)$.*

Proof. We validate that for each C_i^D we have $I^D \models_{\mathbf{D}} C_i^D$. Observe that because of how $d^e(D)$ is defined, every clause in $d^e(D)$ contains either no negative definer literal or $\neg D$ is the only negative definer literal (no clauses with more than one negative definer literal can be derived). This means, for any two clauses $C_i^D, C_j^D \in d^e(D)$, the side conditions of the rules are satisfied (the union never has more than one negative definer literal). We do the proof by contradiction. Assume i is the smallest i with $I^D \not\models_{\mathbf{D}} C_i^D$.

1. If the maximal literal in C_i^D is of the form A or $\exists r.D'$, then the clause is satisfied due to Step 3 in the construction of I^D, which contradicts our assumption.
2. If the maximal literal in C_i^D is of the form $\neg A$, we have $I^D \not\models \neg A$ and therefore $I^D \models A$. This means there must be a clause C_j^D where A is maximal in C_j^D and $I_j^D \not\models C_j^D \setminus \{A\}$, otherwise A is not added to I^D. But then, due to the resolution rule, we also have a clause $C = (C_i^D \cup C_j^D) \setminus \{A, \neg A\}$, which is also in $d^e(D)$. Since $\prec_C$ is the multiset extension of the ordering between literals, C is smaller than C_i^D, since $\neg A \in C_i^D$ and $\neg A$ is larger than all elements in C ($\neg A$ is maximal in C_i^D).
   Since both $I^D \not\models_{\mathbf{D}} C_j^D \setminus \{A\}$ and $I^D \not\models_{\mathbf{D}} C_i^D \setminus \{\neg A\}$, we have $I^D \not\models_{\mathbf{D}} C$. Because C belongs to $d^e(D)$ and is smaller than C_i^D, there is a $k < i$ with $C = C_k^D$, which contradicts our initial assumption that i is the smallest i with $I^D \not\models_{\mathbf{D}} C_i^D$.
3. If the maximal literal in C_i^D is of the form $\forall r.D'$, we have $I^D \not\models_{\mathbf{D}} C_i^D \setminus \{\forall r.D'\}$ and $I^D \not\models_{\mathbf{D}} \forall r.D'$. The only way the latter can be true is due to a literal $\exists r.D_2 \in I^D$ with $D_2 \not\sqsubseteq_D D'$. If D_2 is not subsumed by D', $\exists r.D_2$ is a counter-example for $\forall r.D'$.
   If $\exists r.D_2 \in I^D$, there must be a clause C_j^D such that the maximal literal in C_j^D is $\exists r.D_2$ and $I_j^D \not\models_{\mathbf{D}} C_j^D$. Because of the role propagation rule, we then also have a clause $C_k^D = (C_i^D \cup C_j^D \cup \{\exists r.D_3\}) \setminus \{\forall r.D', \exists r.D_2\}$, where D_3 represents $D' \sqcap D_2$. In our ordering, $\exists r.D_3$ is smaller than both $\forall r.D'$ and $\exists r.D_2$, and therefore $C_k^D \prec_C C_j^D$, $C_k^D \prec_C C_i^D$ and $k < j < i$. We obtain that $I^D \not\models_{\mathbf{D}} C_k^D$ because (i) $I^D \not\models_{\mathbf{D}} C_i^D \setminus \{\forall r.D'\}$, (ii) $I^D \not\models_{\mathbf{D}} C_j^D \setminus \{\exists r.D_2\}$, and (iii) $I^D \not\models_{\mathbf{D}} \exists r.D_3$ (for else $I_j^D \models_{\mathbf{D}} \exists r.D_2$ as $D_3 \sqsubseteq_D D_2$ and $C_k^D \prec_C C_j^D$, and $\exists r.D_3$ cannot be maximal in any clause larger than C_j^D). However, $I^D \not\models_{\mathbf{D}} C_k^D$ contradicts our initial assumption. □

Based on I^D, we construct the candidate model $\mathcal{I}_c = \langle \Delta^{\mathcal{I}_c}, \cdot^{\mathcal{I}_c} \rangle$ with:

- $\Delta^{\mathcal{I}_c} = \{x_D \mid D \in \mathbf{D}$ and I^D is not empty$\}$,
- for every atomic concept A, $A^{\mathcal{I}_c} = \{x_D \mid A \in I^D\}$, and
- for every role r, $r^{\mathcal{I}_c} = \{(x_{D_1}, x_{D_2}) \mid \exists r.D_2 \in I^{D_1}$ *and* I^{D_2} *is nonempty*$\}$.

Lemma 4. *If* $\bot \notin N_s$*, then* $\mathcal{I}_c$ *is a model of* N_s*.*

Proof. We already established that I^D contains all literal concepts that have to hold in order to satisfy the set of clauses $d^e(D)$. All other clauses in N_s are satisfied by I^D as well, since they are all of the form $\neg D' \sqcup C$, and either $D' \notin I^D$, or $\neg D \sqcup D' \in d^e(D)$, and resolution on D' results in $\neg D \sqcup C \in d^e(D)$.

Observe that I^D only contains literals of the form A or $\exists r.D$, which means we do not have to check satisfaction of literals of the form $\forall r.D$. The candidate model is constructed in such a way that for each nonempty I^D, we have a domain element x_D that satisfies all atomic concepts A in I^D. If I^D is empty, it means that we have a unit clause $\neg D$ which is equivalent to $D \sqsubseteq \bot$, and $D^{\mathcal{I}_c}$ should be empty, which is also ensured by the model construction. Therefore we only have to show that, if I^D is nonempty, every existential role restriction $\exists r.D' \in I^D$ holds in $\mathcal{I}_c$ for x_D as well.

- If $I^{D'}$ is not empty, there is a domain element $x_{D'}$ and $(x_D, x_{D'}) \in r^{\mathcal{I}_c}$. Therefore $\exists r.D'$ is satisfied for x_D.
- If $I^{D'}$ is empty, there is no domain element $x_{D'}$, and $\exists r.D'$ is not satisfied by x_D. This can only be the case if $\neg D' \in N_s$. Since we assume $\exists r.D' \in I^D$, there is a clause C_i^D, where $\exists r.D'$ is the maximal literal and $I_i^D \not\models_{\mathbf{D}} C_i^D$. But then, due to the existential role elimination rule and because $\neg D' \in N_s$, there is also the smaller clause $C = C_i^D \setminus \{\exists r.D'\}$. But if $I^D \models C$, then $\exists r.D'$ is not in I^D, which contradicts our assumption that $\exists r.D' \in I^D$. □

We can now prove refutational soundness and completeness of the decision procedure, i.e. Theorem 2.

Proof. Soundness was already established in Lemma 1. Therefore if $N \vdash \bot$, N is unsatisfiable. Suppose $N \not\vdash \bot$. Then we can construct a model for N_s (N saturated by $RES_{\mathcal{ALC}}$) using the method described above (Lemma 4), and since N_s is equi-satisfiable with N (Lemma 1), N is satisfiable. □

7 Correctness of the Uniform Interpolation Method

In order to prove the correctness of our uniform interpolation method, we have to show that every consequence $C \sqsubseteq D$ in the desired signature is preserved by the uniform interpolant. For Phase 2, we use our decision procedure to show that these consequences are preserved by $\mathit{ELIM}_{\mathit{Res}}(N, A)$. This is done by generating a set of clauses M for each consequence $C \sqsubseteq D$, such that $N \models C \sqsubseteq D$ iff $N \cup M$ is unsatisfiable. We first show that for any clause set M over the desired signature, $N \cup M$ is satisfiable iff $\mathit{ELIM}_{\mathit{Res}}(N, A) \cup M$ is satisfiable.

Lemma 5. *For any concept symbol A and any sets of clauses N and M, such that $A \notin sig(M)$, let N_s be the result of saturating $N \cup M$ and N_s' be the result of saturating $ELIM_{Res}(N, A) \cup M$ using $RES_{\mathcal{ALC}}$. It is possible to create a candidate model for N_s' iff it is possible to create a candidate model for N_s.*

Proof. We define the orderings $\prec_L$ and $\prec_C$ as in the last section with the additional constraint that $\neg B \prec_L A$ for any concept symbol $B \neq A$, A being the concept symbol eliminated in N_s'.

We first point out the following properties of clause sets N saturated using $RES_{\mathcal{ALC}}$ regarding definer symbols D: (i) If $|conj(D)| > 1$ (D is an introduced definer), we have $\neg D \sqcup D_i \in N$ for every $D_i \in conj(D)$ (due to role propagation and possibly subsequent resolution steps). Due to further resolution applications, this implies (ii) for every $D_i \sqsubseteq_D D$, we have $d^e(D) \subseteq d^e(D_i)^{D_i \mapsto D}$, where $d^e(D_i)^{D_i \mapsto D}$ denotes the result of replacing every D_i in $d^e(D_i)$ with D. (iii) There is maximally one definer in $conj(D)$ that occurs under an existential role restriction, and every pair of definers in $conj(D)$ occurs under contexts that allow for their combination via the role propagation rule (role propagation is only applied if at least one literal is a universal role restriction and if the side conditions are not violated). (iv) Every nonempty subset of $conj(D)$ is represented by a definer (this is a consequence of (iii)).

Now, observe that in the proof for Lemma 3 only rule applications on maximal literals and definer literals are needed. Resolution on definer literals is assumed indirectly in the proof by how we define satisfaction for literal sets I^D taking into account $\sqsubseteq_D$. This means it is sufficient to perform inferences on maximal literals or definer literals.

A difference between N_s and N_s' is that N_s' does not contain any clauses using A. If we can show that nevertheless conclusions of resolving on A literals occurring in N_s also occur in N_s', we are done, since in $RES_{\mathcal{ALC}}$ rules are applied unrestricted and in $ELIM_{RES}(N, A)$ only clauses containing A are removed. If A is not crucial in deriving the empty clause, it is safe to remove clauses containing it. If A is crucial, the only derivations we lose when removing clauses containing A are conclusions of inference steps involving A.

For pairs of clauses that do not contain any definers, or that only contain definers that are also in $ELIM_{RES}(N, A)$, their resolvents on A are in N_s' since they are in $ELIM_{RES}(N, A)$. Assume we have a clause $C = \neg D \sqcup C_1 \sqcup C_2 \in N_s$ that is the resolvent of two clauses $\neg D \sqcup C_1 \sqcup A, \neg D \sqcup C_2 \sqcup \neg A \in N_s$, such that $C \notin N_s'$, and assume further that C is the largest clause according to $\prec_C$ with this property. As already mentioned, D cannot be in $ELIM_{RES}(N, A)$, since otherwise C is also in $ELIM_{RES}(N, A)$. Hence, D can only co-occur with A due to resolution on clauses of the form $\neg D \sqcup D_i$, where D_i co-occurs with A. This means there are two clauses $\neg D \sqcup D_1, \neg D \sqcup D_2 \in N_s$, where at least one of D_1 and D_2 co-occurs with A in a clause (observe that $D \sqsubseteq_D D_1$ and $D \sqsubseteq_D D_2$). We have two cases:

1. $\neg D_1 \sqcup C_1 \sqcup A, \neg D_1 \sqcup C_2 \sqcup \neg A \in N_s$. Then $\neg D_1 \sqcup C_1 \sqcup C_2 \in N_s'$ (due to our assumption that C is the largest resolvent not in N_s'), and due to resolution on D_1 we have $C \in N_s'$, which contradicts our assumption that $C \notin N_s'$.

2. $\neg D_1 \sqcup C_1 \sqcup A, \neg D_2 \sqcup C_2 \sqcup \neg A \in N_s$. Since at least one definer is not in $\mathit{ELIM}_{RES}(N, A)$ (otherwise D would be in $\mathit{ELIM}_{RES}(N, A)$ as well), there must be clauses $\neg D_1' \sqcup C_1 \sqcup A$ and $\neg D_1 \sqcup D_1'$. But then, due to (iv), we also have a definer D' representing $D_1' \sqcap D_2$, and $D \sqsubseteq_D D'$. Due to our assumption, $\neg D' \sqcup C_1 \sqcup C_2 \in N_s'$. Due to (ii), we also have $\neg D \sqcup C_1 \sqcup C_2 = C \in N_s'$, thus contradicting our assumption that $C \notin N_s'$. □

Now we can prove Theorem 3, which states that for any concept symbol A and clause set N, $\mathit{ELIM}_{Res}(N, A)$ can be computed in finitely bounded time and preserves all consequences over $\mathit{sig}(N) \setminus \{A\}$.

Proof. The fact that $\mathit{ELIM}_{Res}(N, A)$ can always be computed in finitely bounded time follows from Lemma 2. Since in $\mathit{ELIM}_{Res}(N, A)$ all clauses containing A are removed, all symbols in $\mathit{ELIM}_{Res}(N, A)$ are either definers or in $\mathit{sig}(N) \setminus \{A\}$. Hence we only have to check the second condition of the definition of uniform interpolants: $\mathit{ELIM}_{Res}(N, A) \models C \sqsubseteq D$ iff $N \models C \sqsubseteq D$, for any $\mathcal{ALC}$ concept subsumption not containing A.

$N \models C \sqsubseteq D$ can be proven by showing that $C \sqcap \neg D$ is unsatisfiable in N, or by showing that $N' = N \cup \{\top \sqsubseteq \exists r^*.(C \sqcap \neg D)\} \models \bot$, where r^* is a new role not occurring in N. Set $M = \mathit{clauses}(\{\top \sqsubseteq \exists r^*.(C \sqcap \neg D)\}$. Since $A \notin \mathit{sig}(M)$ and due to Lemma 5, we have $N \cup M \models \bot$ iff $\mathit{ELIM}_{Res}(N, A) \cup M \models \bot$. □

We can now prove the correctness of our method (Theorem 1):

Proof. Because of Theorem 3, Phase 2 of the method computes the clausal representation of the uniform interpolant in finite time, independent of the order in which symbols are processed. The correctness of Phase 3 follows from Theorem 4, which can be proved by easy adaptions of the proofs in [1] and [14]. □

8 Related work

A different method for computing uniform interpolants involving fixpoint operators is presented in [13]. This method is based on computing most general and most specific concepts for the concept symbols to be eliminated. In order to avoid infinite derivations, the derivation graph of this process is checked for cycles and fixpoint operators are introduced where necessary. This method exploits the properties of normalised $\mathcal{EL}$-TBoxes and is therefore not immediately applicable for $\mathcal{ALC}$-TBoxes.

A different approach for the description logic $\mathcal{ALC}$ was published in [18]. In this approach a tableaux calculus is used to incrementally add logical consequences from the input ontology. The uniform interpolant is approximated by replacing symbols outside of the signature by $\top$. Adding more consequences before this replacement approximates the result better, which leads to an incremental approximation of the uniform interpolant. If two succeeding approximations are logically equivalent, the uniform interpolant has been computed. This requires periodically checking for TBox equivalence, which can be expensive if the input ontology is large. Using resolution provides a way to make the computation more goal-oriented, and our definer-based representation of nested formulae facilitates the detection of cyclic structures.

Our approach was influenced by a method for uniform interpolation for modal logic K presented in [10]. Formulae in modal logic K are syntactic variants of $\mathcal{ALC}$ concepts, for which uniform interpolants are always finite. As in our approach, their method is based on a resolution calculus, but no structural transformation is used. For this reason, it uses a more complex resolution framework, which is able to perform inferences on nested formulae. The same method can be used to compute uniform interpolants of $\mathcal{ALC}$ concepts, but cannot be extended to TBoxes without affecting termination.

$\mathcal{ALC}\mu$ concepts are syntactic variants of formulae in the modal μ-calculus, which is multi-modal logic K extended with fixpoint operators. Existence of uniform interpolants in the modal μ-calculus was proven in [4], and in [5] a method to compute uniform interpolants is presented. We have not investigated yet whether this calculus can be used in the context of TBox interpolation.

The method we developed is closely related to two methods developed in the context of second-order quantifier elimination [7]. In second-order quantifier elimination, the aim is to eliminate existentially quantified predicate symbols in order to translate second-order formulae into equivalent formulae in first-order logic. In uniform interpolation the aim is to eliminate symbols as well, even though it is not required that the result is logically equivalent to the corresponding formula in second-order logic. The generalised Ackermann's Lemma is used in the second-order quantifier elimination system DLS [14]. Our resolution-procedure follows a similar principle as the second-order quantifier elimination method SCAN [6]. The idea to use the generalised version of Ackermann's Lemma for quantifier-elimination in description logics was first presented in [17]. There has been some work on second-order quantifier elimination for modal logics [8, 15], but it has not yet been investigated how these methods relate to uniform interpolation in description logics.

9 Conclusion and Future Work

We presented a method for computing uniform interpolants of $\mathcal{ALC}$-ontologies. Since uniform interpolants are not always finitely representable in $\mathcal{ALC}$, our method uses fixpoint operators and expresses interpolants in the description logic $\mathcal{ALC}\mu$. If no fixpoint operators are introduced by our method, the result is actually the uniform interpolant in $\mathcal{ALC}$. In the other cases, we offer ways to approximate the result in $\mathcal{ALC}$. Our method mainly consists of two parts. The first part is based on a set of rules influenced by classical resolution to eliminate concept symbols incrementally. This step introduces new symbols, which are eliminated in the second part of our method exploiting the generalised version of Ackermann's Lemma.

We have a first implementation of our method, using further optimisations like standard redundancy elimination techniques, which we did not present here for space reasons. First experiments with available ontologies look promising in the sense that uniform interpolants can be computed in the majority of cases.

One optimisation we are currently working on is optimal use of fixpoint operators. It is possible to create examples where there is a finite uniform interpolant in $\mathcal{ALC}$, even though our method returns a result with fixpoint operators. This is for example the case if the fixpoint expression is redundant because of a cyclic relation between other concepts in the ontology. Investigating several techniques to deal with this problem is the topic of ongoing research. In the future it would be useful to have a method that only introduces fixpoints if they are strictly necessary, that is, if there is no equivalent representation of the interpolant in pure $\mathcal{ALC}$.

Apart from that, we are currently investigating how further description logic constructs such as inverse roles or nominals can be incorporated into our method. We believe that our method can serve as a basis for uniform interpolation in more expressive description logics, such as for example $\mathcal{ALCHI}$.

References

1. Ackermann, W.: Untersuchungen über das Eliminationsproblem der mathematischen Logik. Mathematische Annalen 110(1), 390–413 (1935)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Handbook of Automated Reasoning, pp. 19–99. Elsevier and MIT Press (2001)
3. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: Proc. IJCAI '99. pp. 84–89. Morgan Kaufmann (1999)
4. D'Agonstino, G., Hollenberg, M.: Uniform interpolation, automata and the modal μ-calculus. In: AiML, vol. 1, pp. 73–84. CSLI Pub. (1998)
5. D'Agostino, G., Lenzi, G.: On modal μ-calculus with explicit interpolants. J. Applied Logic 4(3), 256–278 (2006)
6. Gabbay, D., Ohlbach, H.J.: Quantifier elimination in second-order predicate logic. In: Proc. KR '92. pp. 425–435. Morgan Kaufmann (1992)
7. Gabbay, D.M., Schmidt, R.A., Szałas, A.: Second Order Quantifier Elimination: Foundations, Computational Aspects and Applications. College Publ. (2008)
8. Goranko, V., Hustadt, U., Schmidt, R.A., Vakarelov, D.: SCAN is complete for all Sahlqvist formulae. Proc. RelMiCS 7, LNCS vol. 3051 pp. 149–162 (2004)
9. Grau, B.C., Motik, B.: Reasoning over ontologies with hidden content: The import-by-query approach. J. Artificial Intelligence Research 45, 197–255 (2012)
10. Herzig, A., Mengin, J.: Uniform interpolation by resolution in modal logic. In: Proc. JELIA'08, LNCS vol. 3051. pp. 219–231. Springer (2008)
11. Lutz, C., Piro, R., Wolter, F.: $\mathcal{EL}$-concepts go second-order: Greatest fixpoints and simulation quantifiers. In: Proc. DL '10. pp. 43–54. CEUR-WS.org (2010)
12. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. IJCAI '11. pp. 989–995. AAAI Press (2011)
13. Nikitina, N.: Forgetting in General EL Terminologies. In: Description Logics. Proc. DL '11, CEUR-WS.org (2011)
14. Nonnengart, A., Szałas, A.: A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. In: Logic at Work, pp. 307–328. Springer (1999)

15. Schmidt, R.A.: The Ackermann approach for modal logic, correspondence theory and second-order reduction. J. Appl. Logic 10(1), 52–74 (2012)
16. Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond Horn ontologies. In: Proc. IJCAI '11. pp. 1093–1098. AAAI Press (2011)
17. Szałas, A.: Second-order reasoning in description logics. J. Appl. Non-Classical Logics 16(3-4), 517–530 (2006)
18. Wang, Z., Wang, K., Topor, R., Zhang, X.: Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In: Proc. ECAI '10. pp. 47–52. IOS Press (2010)