# Beth Definability in Expressive Description Logics

**Balder ten Cate**[*]
UC Santa Cruz
btencate@ucsc.edu

**Enrico Franconi**
Free University of Bozen-Bolzano
franconi@inf.unibz.it

**İnanç Seylan**[†]
Free University of Bozen-Bolzano
seylan@inf.unibz.it

## Abstract

The *Beth definability property*, a well-known property from classical logic, is investigated in the context of description logics (DLs): if a general $\mathscr{L}$-TBox implicitly defines an $\mathscr{L}$-concept in terms of a given signature, where $\mathscr{L}$ is a DL, then does there always exist over this signature an explicit definition in $\mathscr{L}$ for the concept? This property has been studied before and used to optimize reasoning in DLs. In this paper a complete classification of Beth definability is provided for extensions of the basic DL $\mathcal{ALC}$ with transitive roles, inverse roles, role hierarchies, and/or functionality restrictions, both on arbitrary and on finite structures. Moreover, we present a tableau-based algorithm which computes explicit definitions of at most double exponential size. This algorithm is optimal because it is also shown that the smallest explicit definition of an implicitly defined concept may be double exponentially long in the size of the input TBox. Finally, if explicit definitions are allowed to be expressed in first-order logic then we show how to compute them in EXPTIME.

## 1 Introduction

We address the Beth definability property [Beth, 1953] in the context of description logics (DLs). The Beth definability property relates two notions of definability in a logic $\mathscr{L}$, implicit definability and explicit definability. Implicit definability is a semantic notion: it asks whether the interpretation of a given $\mathscr{L}$-concept $C$ is fully determined by the universe of discourse and the interpretation of some given predicates $\Sigma$ (concept and/or role names) in all models of a theory (TBox). Explicit definability on the other hand is more syntactic: it asks whether there is some $\mathscr{L}$-concept $D$ over the set of predicates $\Sigma$ that is equivalent to $C$ under $\mathcal{T}$. Clearly, explicit definability implies implicit definability. If the converse holds as well, then the logic $\mathscr{L}$ is said to have the Beth definability property. Since the Beth definability property connects the model-theoretic notion of implicit definability to explicit definability, logics having this property are considered to be well-balanced in terms of their syntax and semantics.

Besides the theoretical interest, there are two useful applications of this property in DLs which are concerned with *rewritings*. The first one is related to extracting an equivalent terminology from a general TBox [Baader *et al.*, 2003; ten Cate *et al.*, 2006]. A terminology consists only of acyclic definitions for concept names and they are of particular interest because reasoning with them is 'easier' than with general TBoxes. For example, satisfiability of an $\mathcal{ALC}$-terminology is a PSPACE-complete problem whereas the same problem for general $\mathcal{ALC}$-TBoxes is EXPTIME-complete [Baader *et al.*, 2003]. The second use case is related to computing the certain answers of a concept query given a database (DB) instance (also referred to as 'DBox' in this context) and a TBox that may speak about more predicates than the DB instance [Seylan *et al.*, 2009]. Here the idea is to find an equivalent rewriting of the original query in terms of the predicates that appear in the DB instance. If such a rewriting exists then determining the certain answers of the query can be done efficiently because the problem is reduced to query answering in relational DBs.

Both use cases involve computing explicit definitions on the basis of implicit definitions. A vital question is what is the complexity of this task, both in terms of the time needed to compute the explicit definitions, and in terms of the size of the explicit definitions obtained. This question was first studied by ten Cate et al. for a weaker Beth definability property which considers only concept names in the signature [ten Cate *et al.*, 2006]. In this paper we are interested in the more general Beth definability property that takes into account role names in the signature. We believe that this is more natural for DLs because in a DL knowledge base, role names are considered to be a part of the signature.

Our contributions in this paper are as follows.

- We obtain a complete classification of the Beth definability property for extensions of $\mathcal{ALC}$ with transitive roles, inverse roles, role hierarchies, and/or functionality restrictions, both on arbitrary structures (BP) and on finite structures (BPF). These results are summarized in Table 1. Note that the finite model property (FMP) of all sublogics of $\mathcal{SHOQ}$ follows from [Lutz *et al.*, 2005]; FMP of all sublogics of $\mathcal{SHIO}_+$ follows from [Duc and Lamolle, 2010]; the failure of FMP in $\mathcal{ALCFI}$ and all its extensions

[*]The author is supported by the NSF grant IIS-0905276.
[†]The author is also affiliated with Universität Bremen.

| $\mathcal{S}$ | $\mathcal{H}$ | $\mathcal{I}$ | $\mathcal{F}$ | FMP | BP | BPF |
|---|---|---|---|---|---|---|
| | | | | + | + | + |
| | | | • | + | + | + |
| | | • | | + | + | + |
| | | • | • | - | + | - |
| | • | | | + | - | - |
| | • | | • | + | - | - |
| | • | • | | + | - | - |
| | • | • | • | - | - | - |
| • | | | | + | + | + |
| • | | | • | + | + | + |
| • | | • | | + | + | + |
| • | | • | • | - | + | - |
| • | • | | | + | - | - |
| • | • | | • | + | - | - |
| • | • | • | | + | - | - |
| • | • | • | • | - | - | - |

Table 1: BP and BPF from $\mathcal{ALC}$ to $\mathcal{SHIF}$

follows from [Baader *et al.*, 2003].

- We present a constructive algorithm based on tableaux to compute explicit definitions in $\mathcal{ALC}$ and all of its considered extensions having the Beth definability property. This algorithm runs in 2-ExpTime and computes in the worst case an explicit definition of double exponential size if the concept is implicitly definable. In this respect, the algorithm is optimal because we also show that the smallest explicit definition of an implicitly defined concept may be double exponentially long in the size of the input TBox for each of these DLs.
- We consider the case where explicit definitions are allowed to be expressed in first-order logic. This is particularly relevant for the use case for computing certain answers of a query given a DB instance and a TBox. We present an algorithm that computes a first-order explicit definition of an implicitly defined concept in single ExpTime for all DLs with BP or BPF.

## 2 Preliminaries

We assume familiarity with the DLs $\mathcal{ALC}$ and $\mathcal{SHIF}$ [Baader *et al.*, 2003; Horrocks *et al.*, 2000]. $\mathcal{SHIF}$ is the theoretical basis of the Web Ontology Language OWL-Lite [Horrocks *et al.*, 2003]. In what follows, we will freely use the terminology and notation from [Horrocks *et al.*, 2000]. However for convenience of exposition, we assume role inclusion axioms (RIAs) to be part of the TBox instead of a separate role hierarchy.

We will use signatures often in our definitions. The full signature of $\mathcal{SHIF}$ consists of $N_C$, $N_R$, and $N_{R^+} \subseteq N_R$. Here $N_C$ and $N_R$ are countably infinite and mutually disjoint sets of *concept names* and *role names*, respectively; and $N_{R^+}$ is a countably infinite set of *transitive role names*. By a *predicate*, we mean an element of $N_C$ or $N_R$. For all $R \in N_R$, $R^-$ denotes its *inverse*. The set of *roles* is defined as $N_R \cup \{R^- \mid R \in N_R\}$. The function $\mathsf{Inv}$ over roles is defined as $\mathsf{Inv}(R) = R^-$ if $R \in N_R$, and $\mathsf{Inv}(R) = S$ if $R = S^-$ for an $S \in N_R$. The boolean function $\mathsf{Trans}$ with domain roles is defined such that $\mathsf{Trans}(R) = 1$ if and only if $R \in N_{R^+}$ or $\mathsf{Inv}(R) \in N_{R^+}$.

In this paper, we will be considering several DLs that are fragments of $\mathcal{SHIF}$. Following the standard naming scheme, letters in the name of a language indicate the constructors supported in that language. $\mathcal{I}$ stands for inverse roles and without its presence the set of roles is equal to the set of role names, i.e., $N_R$. $\mathcal{S}$ stands for transitive role names and without its presence the set of transitive role names $N_{R^+}$ is assumed to be empty. $\mathcal{F}$ stands for allowing $\leq 1R$ in the concept language for every role $R$. $\mathcal{H}$ stands for RIAs and without its presence the TBoxes do not contain any RIAs. The 'vanilla' DL without any of these constructors is $\mathcal{ALC}$. $\mathcal{ALC}$ with transitive role names is denoted as $\mathcal{S}$ but we will sometimes use $\mathcal{ALCS}$ for convenience. As usual transitive roles and roles with transitive sub-roles are not allowed in concepts of the form $\leq 1R$ in logics with $\mathcal{S}$ and $\mathcal{F}$ because of decidability concerns [Horrocks *et al.*, 2000]. Finally, we use the symbol $\mathscr{L}$ as a placeholder for any of these DLs.

For an $\mathscr{L}$-concept $C_0$, the set $\mathsf{sub}(C_0)$ consists of $C_0$ and all its subconcepts. For a concept $C_0$ and a TBox $\mathcal{T}$, $\mathsf{rol}(C_0, \mathcal{T})$ denotes the set of roles occurring in $C_0$ or $\mathcal{T}$; and $\mathsf{sig}(C_0, \mathcal{T})$ denotes the set of concept and role names occurring in $C_0$ or $\mathcal{T}$, i.e., the signature of $C_0$ and $\mathcal{T}$.

The Beth definability property, in the general sense, has been first shown to hold for first-order logic [Beth, 1953]. Beth definability comes in different flavors and the one we are interested in is related more to *projective Beth definability*. The projective version is known be stronger than Beth's original formulation [Hoogland, 2001]. Beth definability for first-order fragments used in database theory, e.g., conjunctive queries, can be used as a framework for query rewriting using exact views [Nash *et al.*, 2010]. Besides the treatment of the topic in first-order logic, Lang and Marquis, also motivated from AI, study the propositional variant [Lang and Marquis, 2008]. Beth definability has been extensively studied for modal logics [Gabbay and Maksimova, 2005]. Because of the presence of TBoxes, the version of the Beth definability property we are interested in is for the global consequence relation in modal logics [Goranko and Otto, 2007].

Constructive methods to show Beth definability in DLs have been previously studied in [ten Cate *et al.*, 2006; Seylan *et al.*, 2009; 2010]. These papers also present some results on the size of explicit definitions that can be obtained for implicitly defined concepts. [ten Cate *et al.*, 2006] establish a single exponential lower bound and a triple exponential upper bound for $\mathcal{ALC}$. It is not hard to see that the lower bound proof of [ten Cate *et al.*, 2006] carries to the Beth definability property we consider; and the matching single exponential upper bound on the size of explicit definitions was established in [Seylan *et al.*, 2010]. Here the notion of the 'size' of a concept $C$ is subtle and one can define it either as (i) the number of occurrences of symbols needed to write $C$, or as (ii) the cardinality of the set of $C$'s subconcepts, i.e., $\mathsf{sub}(C)$. For example the algorithm of [Seylan *et al.*, 2010] implies a double exponential upper bound when one uses (i) to measure the size of a concept instead of (ii). Therefore a matching lower bound on the size of explicit definitions was an open problem for the case of (i). Because of this subtlety, we fix what we mean by size for the rest of the paper.

**Definition 2.1.** *The* size of a $\mathscr{L}$-concept $C_0$, *written* $|C_0|$, *is the number of occurrences of symbols needed to write* $C_0$. *The* size of a $\mathscr{L}$-TBox $\mathcal{T}$, *written* $|\mathcal{T}|$, *is defined analogously.*

In the rest of the section, we introduce notions that are related to Beth definability.

**Definition 2.2 (Implicit definability).** *Let* $C$ *be an* $\mathscr{L}$*-concept,* $\mathcal{T}$ *a* $\mathscr{L}$*-TBox, and* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$. $C$ *is* implicitly definable *from* $\Sigma$ *under* $\mathcal{T}$ *if and only if for any two models* $\mathcal{I}$ *and* $\mathcal{J}$ *of* $\mathcal{T}$, *if*

- $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ *and*
- *for all* $P \in \Sigma$, $P^{\mathcal{I}} = P^{\mathcal{J}}$

*then* $C^{\mathcal{I}} = C^{\mathcal{J}}$.

In other words, given a TBox, a concept $C$ is implicitly definable if the set of all its instances depends only on the extension of the predicates in $\Sigma$. *Deciding implicit definability in* $\mathscr{L}$ means, given an $\mathscr{L}$-concept $C$, $\mathscr{L}$-TBox $\mathcal{T}$, and a set of predicates $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$, to check whether $C$ is implicitly definable from $\Sigma$ under $\mathcal{T}$. For every predicate $P \in \mathsf{sig}(C, \mathcal{T}) \setminus \Sigma$, introduce a new predicate $P'$ which is not in $\mathsf{sig}(C, \mathcal{T})$. Now let $\widetilde{C}$ ($\widetilde{\mathcal{T}}$) be the concept (TBox) obtained by replacing every occurrence of a predicate $P \notin \Sigma$ in $C$ ($\mathcal{T}$) by $P'$. The following lemma provides an alternative, more syntactic definition of implicit definability. In particular, it reduces implicit definability in $\mathscr{L}$ to the entailment problem in $\mathscr{L}$.

**Lemma 2.3.** *Let* $C$ *be a* $\mathscr{L}$*-concept,* $\mathcal{T}$ *be a* $\mathscr{L}$*-TBox, and* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$. *Then* $C$ *is implicitly definable from* $\Sigma$ *under* $\mathcal{T}$ *if and only if* $\mathcal{T} \cup \widetilde{\mathcal{T}} \models C \equiv \widetilde{C}$.

It is also possible to reduce TBox-unsatisfiability to implicit definability.

**Lemma 2.4.** *Let* $\mathcal{T}$ *be a* $\mathscr{L}$*-TBox and let* $A_0$ *be a concept name that does not appear in* $\mathcal{T}$. *Then* $\mathcal{T}$ *is unsatisfiable if and only if* $A_0$ *is implicitly definable from* $\mathsf{sig}(\mathcal{T})$ *under* $\mathcal{T}$.

By Lemmas 2.3 and 2.4, the following theorem follows immediately, given that the concept satisfiability problem for each of these DLs is EXPTIME-complete.

**Theorem 2.5.** *In* $\mathcal{ALC}$ *and any of its extensions with constructors from* $\{\mathcal{S}, \mathcal{H}, \mathcal{I}, \mathcal{F}\}$, *implicit definability is* EXPTIME*-complete.*

If a concept is implicitly definable from $\Sigma$, then it may be possible to find an expression using only predicates in $\Sigma$ whose instances are the same as in the original concept: this would be its explicit definition.

**Definition 2.6 (Explicit definability).** *Let* $C$ *be a* $\mathscr{L}$*-concept,* $\mathcal{T}$ *a* $\mathscr{L}$*-TBox, and* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$. $C$ *is* explicitly definable *from* $\Sigma$ *under* $\mathcal{T}$ *if and only if there is some* $\mathscr{L}$*-concept* $D$ *such that* $\mathcal{T} \models C \equiv D$ *and* $\mathsf{sig}(D) \subseteq \Sigma$. *Such a* $D$ *is called an* explicit definition *of* $C$ *from* $\Sigma$ *under* $\mathcal{T}$.

**Proposition 2.7.** *Let* $C$ *be a* $\mathscr{L}$*-concept,* $\mathcal{T}$ *a* $\mathscr{L}$*-TBox, and* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$. *If* $C$ *is explicitly definable from* $\Sigma$ *under* $\mathcal{T}$ *then* $C$ *is implicitly definable from* $\Sigma$ *under* $\mathcal{T}$.

**Definition 2.8 (Beth definability property).** $\mathscr{L}$ *has the* Beth definability property *(BP) if for all* $\mathscr{L}$*-concepts* $C$, *all* $\mathscr{L}$*-TBoxes* $\mathcal{T}$, *and all signatures* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$, *if* $C$ *is implicitly definable from* $\Sigma$ *under* $\mathcal{T}$ *then* $C$ *is explicitly definable from* $\Sigma$ *under* $\mathcal{T}$.

Clearly, we can restrict the role names that are allowed to appear in our explicit definitions by putting these role names into $\Sigma$. Let us call the version of BP that does not restrict role names but only concept names occurring in the explicit definitions the *concept name BP* (CBP). That is in CBP, we look for explicit definitions over subsets of $\Sigma \cup N_R$.

A logic may lack the finite model property (FMP) and in some cases it is more natural to consider only finite models. This is also the approach taken in DB theory. For example, in a DL KB with a DB instance, the purpose of the DB instance is to fix the extension of some predicates [Seylan *et al.*, 2009]. Some DLs lack FMP and because of this the TBox may enforce the DB instance to be infinite, i.e., the DB instance can not be fixed.

**Definition 2.9 (Finite model property).** *An interpretation is said to be* finite *if it has a finite domain. A DL* $\mathscr{L}$ *is said to have the* finite model property *(FMP) if for every* $\mathscr{L}$*-concept* $C$ *and every* $\mathscr{L}$*-TBox* $\mathcal{T}$, *if* $C$ *is satisfiable w.r.t.* $\mathcal{T}$ *then there is some finite interpretation* $\mathcal{I}$ *such that* $\mathcal{I}$ *is a model of* $\mathcal{T}$ *and* $C^{\mathcal{I}} \neq \emptyset$.

A relevant question in this case is if the Beth definability property holds when one restricts attention to finite models. For example, Beth definability, when restricted to finite models, fails in first-order logic [Hoogland, 2001] although it holds in the unrestricted case. In this paper, we therefore also investigate BP restricted to finite interpretations. We call this version of the problem *Beth definability property in the finite* (BPF). Instead of redefining the relevant notions for BPF, we assume that all our definitions are the same except that we replace the word 'model' with 'finite model' and the symbol $\models$ with $\models_f$, where $\models_f$ considers only finite models. We call this version of implicit (explicit) definability f-implicit (resp. f-explicit) definability when we want to be precise.

If $\mathscr{L}$ has FMP then BP are BPF are equivalent because of Lemma 2.3 and the fact that $\models$ coincides with $\models_f$. Hence it only makes sense to consider BPF in logics without FMP.

## 3 Constructive Interpolation with Tableaux

This section provides a constructive proof of an interpolation property which will be the essential part of the proof of BP in Section 4. Resorting to interpolation to show the Beth definability property in a logic has been a standard technique since Craig's seminal paper [Craig, 1957]. We start by defining what we mean by an interpolant and then state the main result of this section.

**Definition 3.1 (Interpolant).** *Let* $C, D$ *be* $\mathscr{L}$*-concepts and let* $\mathcal{T}_1, \mathcal{T}_2$ *be* $\mathscr{L}$*-TBoxes such that* $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq D$. *An* $\mathscr{L}$*-concept* $I$ *is called an* interpolant *of* $C$ *and* $D$ *under* $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ *if the following conditions hold:*

- $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{T}_1) \cap \mathsf{sig}(D, \mathcal{T}_2)$,
- $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq I$, *and*
- $\mathcal{T}_1 \cup \mathcal{T}_2 \models I \sqsubseteq D$.

**Theorem 3.2.** *Let* $\mathscr{L}$ *be* $\mathcal{ALC}$ *or any of its extensions with constructors from* $\{\mathcal{S}, \mathcal{I}, \mathcal{F}\}$. *For all* $\mathscr{L}$*-concepts* $C_1, C_2$ *and all* $\mathscr{L}$*-TBoxes* $\mathcal{T}_1, \mathcal{T}_2$, *if* $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ *then there exists an interpolant of* $C_1$ *and* $C_2$ *under* $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ *that can be computed in time double exponential in* $|\mathcal{T}_1| + |\mathcal{T}_2| + |C_1| + |C_2|$.

The proof of Theorem 3.2 consists of two stages. First Theorem 3.2 is shown to hold for $\mathcal{ALC}$ and $\mathcal{ALCF}$ directly using a worst-case optimal tableau algorithm in the style of Goré and Nyugen [Goré and Nguyen, 2007]. Then we show that Theorem 3.2 holds for extensions of $\mathcal{ALC}$ and $\mathcal{ALCF}$ via satisfiability and signature preserving reductions to $\mathcal{ALC}$ and $\mathcal{ALCF}$.

It will be convenient for us to assume that concepts are in *negation normal form* (NNF) in the rest of the section. This is w.l.o.g. if we take e.g. $\forall R.C$ not as an abbreviation of $\neg\exists R.\neg C$ but as a primitive constructor of the language. The NNF of the complement of a concept $C$ is written $\dot{\neg}C$. Again w.l.o.g., we assume that a TBox consists only of axioms of the form $\top \sqsubseteq C$.

**Definition 3.3.** *Let $C_0$ be an $\mathcal{ALCF}$-concept ($\mathcal{ALCFI}$-concept) and let $\mathcal{T}$ be an $\mathcal{ALCF}$-TBox ($\mathcal{ALCFI}$-TBox). The* concept closure $\mathsf{cl}(C_0, \mathcal{T})$ *of $C_0$ and $\mathcal{T}$ is the smallest set of concepts satisfying the following conditions:*

- $C_0 \in \mathsf{cl}(C_0, \mathcal{T})$;
- *if* $\top \sqsubseteq C \in \mathcal{T}$ *then* $C \in \mathsf{cl}(C_0, \mathcal{T})$;
- *if* $C \in \mathsf{cl}(C_0, \mathcal{T})$ *and* $D \in \mathsf{sub}(C)$ *then* $D \in \mathsf{cl}(C_0, \mathcal{T})$;
- $\{\leq 1R, \exists R.C\} \subseteq \mathsf{cl}(C_0, \mathcal{T})$ *then* $\forall R.C \in \mathsf{cl}(C_0, \mathcal{T})$.

For the rest of this section, fix two $\mathcal{ALCF}$-concepts $C_0$ and $D_0$, and three $\mathcal{ALCF}$-TBoxes $\mathcal{T}_\mathbf{l}$, $\mathcal{T}_\mathbf{r}$, and $\mathcal{T}$ such that $\mathcal{T}_\mathbf{l} \cup \mathcal{T}_\mathbf{r} = \mathcal{T}$. $\mathbf{l}$ stand for *left* and $\mathbf{r}$ for *right* and it is a naming scheme adopted from [Fitting, 1996]. It will allows us to identify from which TBox ($\mathcal{T}_\mathbf{l}$ or $\mathcal{T}_\mathbf{r}$) or concept ($C_0$ or $D_0$) an inference is made.

A *biased concept* is an expression of the form $C^\lambda$, where $C$ is an $\mathcal{ALCF}$-concept and $\lambda \in \{\mathbf{l}, \mathbf{r}\}$ is a *bias*. Two relevant biased concept closures $\mathsf{cll}$ and $\mathsf{clr}$ are defined as follows.

$$\mathsf{cll} = \{C^\mathbf{l} \mid C \in \mathsf{cl}(C_0, \mathcal{T}_\mathbf{l})\} \text{ and } \mathsf{clr} = \{C^\mathbf{r} \mid C \in \mathsf{cl}(\dot{\neg}D_0, \mathcal{T}_\mathbf{r})\}.$$

We use the Greek letters $\lambda, \kappa$ to denote a bias.

**Definition 3.4.** *Let $\Phi \subseteq \mathsf{cll} \cup \mathsf{clr}$. Then*

- $(C_1 \sqcap C_2)^\lambda$ *is an* $\sqcap$-burden *of* $\Phi$ *iff* $(C_1 \sqcap C_2)^\lambda \in \Phi$ *and* $\{(C_1)^\lambda, (C_2)^\lambda\} \not\subseteq \Phi$;
- $(C_1 \sqcup C_2)^\lambda$ *is an* $\sqcup$-burden *of* $\Phi$ *iff* $(C_1 \sqcup C_2)^\lambda \in \Phi$ *and* $\{(C_1)^\lambda, (C_2)^\lambda\} \cap \Phi = \emptyset$;
- $(\leq 1R)^\lambda$ *is an* $\leq$ 1-burden *of* $\Phi$ *iff* $(\leq 1R)^\lambda \in \Phi$ *and* $\{(\forall R.C)^\kappa \mid (\exists R.C)^\kappa \in \Phi\} \not\subseteq \Phi$;
- $(\exists R.C)^\lambda$ *is an* $\exists$-burden *of* $\Phi$ *iff* $(\exists R.C)^\lambda$ *is in* $\Phi$;
- $(\geq 2R)^\lambda$ *is an* $\geq$ 2-burden *of* $\Phi$ *iff* $(\geq 2R)^\lambda$ *is in* $\Phi$.

*A* burden *of* $\Phi$ *is any type of burden from above.*

**Definition 3.5.** *Let $\Phi \subseteq \mathsf{cll} \cup \mathsf{clr}$, $C^\lambda$ be a burden of $\Phi$, and $S = \{D^\mathbf{l} \mid \top \sqsubseteq D \in \mathcal{T}_\mathbf{l}\} \cup \{D^\mathbf{r} \mid \top \sqsubseteq D \in \mathcal{T}_\mathbf{r}\}$. Then $\Psi \subseteq \mathsf{cll} \cup \mathsf{clr}$ is called the $C^\lambda$-*relief *of $\Phi$ if*

- $C = (C_1 \sqcap C_2)^\lambda$ *and* $\Psi = \{(C_1)^\lambda, (C_2)^\lambda\} \cup \Phi$;
- $C = (C_1 \sqcup C_2)^\lambda$ *and either* $\Psi = \Phi \cup \{(C_1)^\lambda\}$ *or* $\Psi = \Phi \cup \{(C_2)^\lambda\}$;
- $C = (\leq 1R)^\lambda$ *and* $\Psi = \Phi \cup \{(\forall R.C)^\kappa \mid (\exists R.C)^\kappa \in \Phi\}$;
- $C = (\exists R.C)^\lambda$ *and* $\Psi = \{C^\lambda\} \cup \{D^\kappa \mid (\forall R.D)^\kappa \in \Phi\} \cup S$;
- $C = (\geq 2R)^\lambda$ *and* $\Psi = \{D^\kappa \mid (\forall R.D)^\kappa \in \Phi\} \cup S$.

For all $\Phi \subseteq \mathsf{cll} \cup \mathsf{clr}$, we define

$$\Phi(\mathbf{l}) = \{C \mid C^\mathbf{l} \in \Phi \cap \mathsf{cll}\} \text{ and } \Phi(\mathbf{r}) = \{C \mid C^\mathbf{r} \in \Phi \cap \mathsf{clr}\}.$$

| | |
|---|---|
| **The $\mathsf{R}_\sqcap$ rule** | |
| *Condition:* | $(C_1 \sqcap C_2)^\lambda$ is an $\sqcap$-burden of $g.\mathsf{content}$. |
| *Action:* | $g'.\mathsf{content} \leftarrow \Phi$, where $\Phi$ is the $(C_1 \sqcap C_2)^\lambda$-relief of $g.\mathsf{content}$. |
| **The $\mathsf{R}_\sqcup$ rule** | |
| *Condition:* | $(C_1 \sqcup C_2)^\lambda$ is an $\sqcup$-burden of $g.\mathsf{content}$. |
| *Action:* | $g_1.\mathsf{content} \leftarrow \Phi_1$ and $g_2.\mathsf{content} \leftarrow \Phi_2$, where $\Phi_1, \Phi_2$ are $(C_1 \sqcup C_2)^\lambda$-reliefs of $\Phi$. |
| **The $\mathsf{R}_{\leq 1}$ rule** | |
| *Condition:* | $(\leq 1R)^\lambda$ is an $\leq$ 1-burden of $g.\mathsf{content}$. |
| *Action:* | $g'.\mathsf{content} \leftarrow \Phi$, where $\Phi$ is the $(\leq 1R)^\lambda$-relief of $g.\mathsf{content}$. |
| **The $\mathsf{R}_\exists$ rule** | |
| *Condition:* | $\Phi = \{(C_1)^{\lambda_1}, \ldots, (C_n)^{\lambda_n}\}$ such that $C^\lambda \in \Phi$ iff $C^\lambda$ is an $\exists$- or $\geq 2$-burden of $g.\mathsf{content}$. |
| *Action:* | For $1 \leq i \leq n$, $g_i.\mathsf{content} \leftarrow \Phi_i$, where $\Phi_i$ is the $(C_i)^{\lambda_i}$-relief of $g.\mathsf{content}$. |

Figure 1: Tableau expansion rules for $\mathcal{ALCF}$.

$\Phi(\lambda)$ is a shorthand for $\Phi(\mathbf{l}) \cup \Phi(\mathbf{r})$. In the following the *signature of a set of $\mathcal{ALCF}$-concepts* $S$ will be of concern. We define $\mathsf{sig}(S) = \bigcup_{C \in S} \mathsf{sig}(C)$.

A *biased $\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau* ($\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau for short) is a directed graph $\langle \mathcal{V}, \mathcal{E}\rangle$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. We associate three different labels to nodes in $\mathcal{V}$.

1. $\mathsf{content} : \mathcal{V} \to 2^{\mathsf{cll} \cup \mathsf{clr}}$,
2. $\mathsf{type} : \mathcal{V} \to \{\texttt{and-node}, \texttt{or-node}\}$,
3. $\mathsf{status} : \mathcal{V} \to \{\texttt{sat}, \texttt{unsat}\}$,
4. $\mathsf{int} : \mathcal{V} \to \mathcal{ALCF}$

The function of these labels are explained when they are used.

We say that a node $g$ in a tableau contains a *clash* if and only if either one of the following holds.

- $\bot^\lambda \in g.\mathsf{content}$,
- $\{A^\lambda, (\neg A)^\kappa\} \subseteq g.\mathsf{content}$,
- $\{(\leq 1R)^\lambda, (\geq 2R)^\kappa\} \subseteq g.\mathsf{content}$.

The *tableau expansion rules* given in Figure 1 expand a tableau by making use of the semantics of concepts. A rule is said to be *applicable* to a node $g$ if and only if its condition is satisfied in $g$, no rule was applied to $g$ before, and $g$ does not contain a clash. In order to guarantee a finite expansion, we use *proxies* in the following way. Whenever a rule creates a new node $g'$ from $g$, before attaching the edge $\langle g, g'\rangle$ to $\mathcal{E}$, the tableau is searched for a node $g'' \in \mathcal{V}$ such that $g'.\mathsf{content} = g''.\mathsf{content}$. If such a $g''$ is found then the edge $\langle g, g''\rangle$ is added to $\mathcal{E}$ and $g'$ is discarded.

We are interested in deciding $\mathcal{T} \models C_0 \sqsubseteq D_0$. The tableau algorithm consists of two phases. The *first phase* starts with the *initial $\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau* $\mathbf{T} = \langle\{g_0\}, \emptyset\rangle$, where $g_0.\mathsf{content} = \{(C_0)^\mathbf{l}, (\dot{\neg}D_0)^\mathbf{r}\} \cup \{E^\mathbf{l} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{l}\} \cup \{E^\mathbf{r} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{r}\}$. $\mathbf{T}$ is then expanded by repeatedly applying the tableau expansion rules in such a way that if more than one rule is applicable to a node at the same time then the first applicable rule in the list $[\mathsf{R}_\sqcap, \mathsf{R}_\sqcup, \mathsf{R}_{\leq 1}, \mathsf{R}_\exists]$ is chosen. If $\mathsf{R}_\sqcup$ has been applied to a node $g$ then we set $g.\mathsf{type} \leftarrow \texttt{or-node}$, and if some rule other than $\mathsf{R}_\sqcup$ has been applied to $g$ we set $g.\mathsf{type} \leftarrow \texttt{and-node}$. The first phase continues as long as some rule is applicable to $\mathbf{T}$.

A $\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau is called *complete* if and only if it is the output of the first phase of the tableau algorithm.

Let $\mathbf{T}$ be a complete $\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau. The purpose of the *second phase* of the tableau algorithm, i.e., Algorithm 1, is to assign a *status* to every node in $\mathbf{T}$ and to calculate a concept $\mathsf{int}(g)$ for every $g \in \mathcal{V}$ with an unsatisfiable content. To this aim, it uses the interpolant calculation rules which are presented in Figure 2.

**Algorithm 1** Second phase of the tableau algorithm

**Propagate**: Do

- $\mathsf{done} \leftarrow \mathsf{true}$.
- For every $g \in \mathcal{V}$ with $g.\mathsf{status} \neq \mathtt{unsat}$:
  - if $g$ contains a clash then (i) $g.\mathsf{status} \leftarrow \mathtt{unsat}$, (ii) apply one of $\{\mathsf{C}^{\mathbf{l}}_{\bot}, \mathsf{C}^{\mathbf{r}}_{\bot}, \mathsf{C}^{\mathbf{ll}}_{\neg}, \mathsf{C}^{\mathbf{rr}}_{\neg}, \mathsf{C}^{\mathbf{lr}}_{\neg}, \mathsf{C}^{\mathbf{rl}}_{\neg}\}$, one whose condition is satisfied, (iii) $\mathsf{done} \leftarrow \mathsf{false}$.
  - if $g.\mathsf{type} = \mathtt{and\text{-}node}$ and there is some direct successor $g'$ of $g$ such that $g'.\mathsf{status} = \mathtt{unsat}$ then (i) $g.\mathsf{status} \leftarrow \mathtt{unsat}$, (ii) apply one of $\{\mathsf{C}_{\sqcap}, \mathsf{C}^{\mathbf{l}R}_{\leq 1}, \mathsf{C}^{\mathbf{r}R}_{\leq 1}, \mathsf{C}^{\mathbf{l}R}_{\leq 1}, \mathsf{C}^{\mathbf{r}R}_{\leq 1}, \mathsf{C}^{\mathbf{l}R}_{\exists}, \mathsf{C}^{\mathbf{r}R}_{\exists}, \mathsf{C}^{\mathbf{l}R}_{\exists}, \mathsf{C}^{\mathbf{r}R}_{\exists}\}$, one whose condition is satisfied, (iii) $\mathsf{done} \leftarrow \mathsf{false}$.
  - if $g.\mathsf{type} = \mathtt{or\text{-}node}$ and for all direct successors $g'$ of $g$ we have $g'.\mathsf{status} = \mathtt{unsat}$ then (i) $g.\mathsf{status} \leftarrow \mathtt{unsat}$, (ii) apply one of $\{\mathsf{C}^{\mathbf{l}}_{\sqcup}, \mathsf{C}^{\mathbf{r}}_{\sqcup}\}$, one whose condition is satisfied, (iii) $\mathsf{done} \leftarrow \mathsf{false}$.

while $\mathsf{done} = \mathsf{false}$.
**Assign**:

For every $g \in \mathcal{V}$ with $g.\mathsf{status} \neq \mathtt{unsat}$, $g.\mathsf{status} \leftarrow \mathtt{sat}$.

Let $\mathbf{T} = \langle \mathcal{V}, \mathcal{E}\rangle$ be a complete $\langle C_0 \sqsubseteq D_0, \mathcal{T}\rangle$-tableau which is an output of the second phase. $\mathbf{T}$ is said to be *open* if and only if $g_0.\mathsf{status} = \mathtt{sat}$; and it is said to be *closed* if and only if $g_0.\mathsf{status} = \mathtt{unsat}$. If $\mathbf{T}$ is determined to be open after the second phase, then the tableau algorithm returns "$\mathcal{T} \not\models C_0 \sqsubseteq D_0$", otherwise it returns "$\mathcal{T} \models C_0 \sqsubseteq D_0$".

The tableau algorithm is a decision procedure for concept subsumption (and satisfiability) in $\mathcal{ALCF}$. Although a complete tableau can be constructed in ExpTime, the interpolants calculated in the second phase may be double exponentially long. Hence the algorithm runs in 2-ExpTime. If $\mathcal{T} \models C_0 \sqsubseteq D_0$ then one can show that $\mathsf{int}(g_0)$ is the interpolant we are looking for and this is the idea behind the proof of the following theorem.

**Theorem 3.6.** *For all $\mathcal{ALCF}$-concepts $C, D$ and all $\mathcal{ALCF}$-TBoxes $\mathcal{T}_1, \mathcal{T}_2$ if $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq D$ then there exists an interpolant of $C$ and $D$ under $\langle \mathcal{T}_1, \mathcal{T}_2\rangle$ that can be computed in time double exponential in $|\mathcal{T}_1| + |\mathcal{T}_2| + |C| + |D|$.*

Now Theorem 3.2, which extends Theorem 3.6 to more logics, can be shown to hold as follows. For $\mathcal{ALC}$, we have that the tableau algorithm for $\mathcal{ALCF}$ also decides concept satisfiability w.r.t. a TBox in $\mathcal{ALC}$. Using the fact that in any execution of the algorithm for input in $\mathcal{ALC}$, one can observe that $\mathsf{R}_{\leq 1}$ will never be applied and the interpolant calculation rules from Figure 2 will never produce concepts of the form $\leq 1R$ or $\geq 2R$. Hence the output interpolant, if there is any, will be in $\mathcal{ALC}$. For more expressive logics, the proof is more involved and requires the following reductions.

**Definition 3.7.** *Let $C_0$ be a $\mathcal{SIF}$-concept and $\mathcal{T}$ be a $\mathcal{SIF}$-TBox. Then $\tau_{\mathcal{S}}(C_0, \mathcal{T})$ is defined as the $\mathcal{ALCFI}$-TBox $\mathcal{T} \cup \mathcal{T}'$, where*

$$\mathcal{T}' = \{\forall R.C \sqsubseteq \forall R.\forall R.C \mid \forall R.C \in \mathsf{cl}(C_0, \mathcal{T}) \text{ and } \mathsf{Trans}(R)\}.$$

[Tobies, 2001] shows that a $\mathcal{SIF}$-concept $C_0$ is satisfiable w.r.t. a $\mathcal{SIF}$-TBox $\mathcal{T}$ if and only if $C_0$ is satisfiable w.r.t. the

**The $\mathsf{C}^{\mathbf{l}}_{\bot}$ rule**
*Condition:* $\bot^{\mathbf{l}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \bot$
**The $\mathsf{C}^{\mathbf{r}}_{\bot}$ rule**
*Condition:* $\bot^{\mathbf{r}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \top$
**The $\mathsf{C}^{\mathbf{ll}}_{\neg}$ rule**
*Condition:* $\{A^{\mathbf{l}}, (\neg A)^{\mathbf{l}}\} \subseteq g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \bot$
**The $\mathsf{C}^{\mathbf{rr}}_{\neg}$ rule**
*Condition:* $\{A^{\mathbf{r}}, (\neg A)^{\mathbf{r}}\} \subseteq g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \top$
**The $\mathsf{C}^{\mathbf{lr}}_{\neg}$ rule**
*Condition:* $\{A^{\mathbf{l}}, (\neg A)^{\mathbf{r}}\} \subseteq g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow A$
**The $\mathsf{C}^{\mathbf{rl}}_{\neg}$ rule**
*Condition:* $\{A^{\mathbf{r}}, (\neg A)^{\mathbf{l}}\} \subseteq g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \neg A$
**The $\mathsf{C}_{\sqcap}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(C_1 \sqcap C_2)^{\lambda}$-relief of $g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g')$.
**The $\mathsf{C}^{\mathbf{l}}_{\sqcup}$ rule**
*Condition:* $g_1.\mathsf{content}, g_2.\mathsf{content}$ are $(C_1 \sqcup C_2)^{\mathbf{l}}$-reliefs of $g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g_1) \sqcup \mathsf{int}(g_2)$.
**The $\mathsf{C}^{\mathbf{r}}_{\sqcup}$ rule**
*Condition:* $g_1.\mathsf{content}, g_2.\mathsf{content}$ are $(C_1 \sqcup C_2)^{\mathbf{r}}$-reliefs of $g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g_1) \sqcap \mathsf{int}(g_2)$.
**The $\mathsf{C}^{\mathbf{l}\not R}_{\leq 1}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\leq 1R)^{\mathbf{l}}$-relief of $g.\mathsf{content}$ and there is no biased concept of the form $(\exists R.C)^{\mathbf{r}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g')$.
**The $\mathsf{C}^{\mathbf{r}\not R}_{\leq 1}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\leq 1R)^{\mathbf{r}}$-relief of $g.\mathsf{content}$ and there is no biased concept of the form $(\exists R.C)^{\mathbf{l}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g')$.
**The $\mathsf{C}^{\mathbf{l}R}_{\leq 1}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\leq 1R)^{\mathbf{l}}$-relief of $g.\mathsf{content}$ and there is some biased concept of the form $(\exists R.C)^{\mathbf{r}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g') \sqcap \leq 1R$.
**The $\mathsf{C}^{\mathbf{r}R}_{\leq 1}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\leq 1R)^{\mathbf{r}}$-relief of $g.\mathsf{content}$ and there is some biased concept of the form $(\exists R.C)^{\mathbf{l}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \mathsf{int}(g') \sqcup \geq 2R$.
**The $\mathsf{C}^{\mathbf{l}\not R}_{\exists}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\exists R.C)^{\mathbf{l}}$- or $(\geq 2R)^{\mathbf{l}}$-relief of $g.\mathsf{content}$, there is no biased concept of the form $(\forall R.D)^{\mathbf{r}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \bot$.
**The $\mathsf{C}^{\mathbf{r}\not R}_{\exists}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\exists R.C)^{\mathbf{r}}$- or $(\geq 2RC)^{\mathbf{r}}$-relief of $g.\mathsf{content}$, there is no biased concept of the form $(\forall R.D)^{\mathbf{l}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \top$.
**The $\mathsf{C}^{\mathbf{l}R}_{\exists}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\exists R.C)^{\mathbf{l}}$- or $(\geq 2R)^{\mathbf{l}}$-relief of $g.\mathsf{content}$, there is some biased concept of the form $(\forall R.D)^{\mathbf{r}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \exists R.\mathsf{int}(g')$.
**The $\mathsf{C}^{\mathbf{r}R}_{\exists}$ rule**
*Condition:* $g'.\mathsf{content}$ is the $(\exists R.C)^{\mathbf{r}}$- or $(\geq 2R)^{\mathbf{r}}$-relief of $g.\mathsf{content}$, there is some biased concept of the form $(\forall R.D)^{\mathbf{l}} \in g.\mathsf{content}$.
*Action:* $\mathsf{int}(g) \leftarrow \forall R.\mathsf{int}(g')$.

Figure 2: Interpolant calculation rules for $\mathcal{ALCF}$.

$\mathcal{ALCFI}$-TBox $\tau_{\mathcal{S}}(C_0, \mathcal{T})$. Using this result, one can prove Lemma 3.8.

**Lemma 3.8.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be $\mathcal{SIF}$-TBoxes and let $C_1, C_2$ be $\mathcal{SIF}$-concepts. Then*

$$\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2 \text{ iff } \tau_{\mathcal{S}}(C_1, \mathcal{T}_1) \cup \tau_{\mathcal{S}}(\dot{\neg} C_2, \mathcal{T}_2) \models C_1 \sqsubseteq C_2.$$

We need a similar reduction to eliminate inverse roles. What is different from the previous reduction is that the signature of the original TBox is slightly modified in the result-

ing TBox. Let $C_0$ be an $\mathcal{ALCFI}$-concept and let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox. For each inverse role $R^- \in \mathsf{rol}(C_0, \mathcal{T})$, introduce a new role name $R^c$ which is not in $\mathsf{rol}(C_0, \mathcal{T})$. Now let $\zeta$ be the total function with domain $\mathsf{cl}(C_0, \mathcal{T})$ such that $\zeta(C)$ is obtained from $C \in \mathsf{cl}(C_0, \mathcal{T})$ by replacing every inverse role $R^-$ occurring in $C$ by $R^c$. We extend $\zeta$ to $\mathcal{T}$ as follows.

$$\zeta(\mathcal{T}) = \{\top \sqsubseteq \zeta(C) \mid \top \sqsubseteq C \in \mathcal{T}\}$$

**Definition 3.9.** *Let $C_0$ be an $\mathcal{ALCFI}$-concept and let $\mathcal{T}$ be an $\mathcal{ALCFI}$-TBox. Then $\tau_{\mathcal{I}}(C_0, \mathcal{T})$ is defined as the $\mathcal{ALCF}$-TBox $\zeta(\mathcal{T}) \cup \mathcal{T}'$, where $\mathcal{T}'$ is equal to the union of the following:*

- $\{\dot\neg\zeta(C) \sqsubseteq \forall R.\exists R^c.\dot\neg\zeta(C) \mid \forall R^-.C \in \mathsf{cl}(C_0, \mathcal{T}),\ R \in N_R\}$,
- $\{\dot\neg\zeta(C) \sqsubseteq \forall R^c.\exists R.\dot\neg\zeta(C) \mid \forall R.C \in \mathsf{cl}(C_0, \mathcal{T}),\ R \in N_R\}$.

[Calvanese *et al.*, 2001] show that an $\mathcal{ALCFI}$-concept $C_0$ is satisfiable w.r.t. an $\mathcal{ALCFI}$-TBox $\mathcal{T}$ if and only if the $\mathcal{ALCF}$-concept $\zeta(C_0)$ is satisfiable w.r.t. the $\mathcal{ALCF}$-TBox $\tau_{\mathcal{I}}(C_0, \mathcal{T})$. Using this result, one can prove Lemma 3.10.

**Lemma 3.10.** *Let $\mathcal{T}_1, \mathcal{T}_2$ be $\mathcal{ALCFI}$-TBoxes and let $C_1, C_2$ be $\mathcal{ALCFI}$-concepts. Then $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ iff*

$$\tau_{\mathcal{I}}(C_1, \mathcal{T}_1) \cup \tau_{\mathcal{I}}(\dot\neg C_2, \mathcal{T}_2) \models \zeta(C_1) \sqsubseteq \zeta(C_2).$$

Now the proof of Theorem 3.2 can be summarized as follows. Given $\mathscr{L}$-TBoxes $\mathcal{T}_1, \mathcal{T}_2$ and $\mathscr{L}$-concepts $C_1, C_2$, we reduce, by Lemma 3.8 and Lemma 3.10, $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ to $\mathcal{T}_1' \cup \mathcal{T}_2' \models C_1' \sqsubseteq C_2'$, where $\mathcal{T}_1', \mathcal{T}_2'$ are $\mathcal{ALC}$- ($\mathcal{ALCF}$-) TBoxes and $C_1', C_2'$ are $\mathcal{ALC}$- ($\mathcal{ALCF}$-) concepts. If $\mathcal{T}_1' \cup \mathcal{T}_2' \models C_1' \sqsubseteq C_2'$ holds then we know by Theorem 3.6 that there is an interpolant $I$ in $\mathcal{ALC}$ ($\mathcal{ALCF}$). It is possible to show, by Lemma 3.8 and Lemma 3.10 again, that $I$ yields an interpolant in $\mathscr{L}$.

[Seylan *et al.*, 2009] also present a procedure for computing interpolants in $\mathcal{ALC}$ in 2-ExpTime. This procedure is based on a non-optimal tableau algorithm. An advantage of our algorithm is that it allows us to compute first-order interpolants in ExpTime by using a succinct representation of concepts assigned to $\mathsf{int}(g)$. This is formalized in Theorem 3.11.

**Theorem 3.11.** *Let $\mathscr{L}$ be $\mathcal{ALC}$ or any of its extensions with constructors from $\{\mathcal{S}, \mathcal{I}, \mathcal{F}\}$. For all $\mathscr{L}$-concepts $C$, $D$ and $\mathscr{L}$-TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$, if $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq D$ then there exists a first-order formula $\varphi(x)$ such that $\varphi(x)$ is equivalent to an interpolant $I$ of $C$ and $D$ under $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$, $\mathsf{sig}(\varphi(x)) \subseteq \mathsf{sig}(I)$, and $\varphi(x)$ can be computed in time single exponential in $|\mathcal{T}_1| + |\mathcal{T}_2| + |C| + |D|$.*

The proof of Theorem 3.11 proceeds along the following lines. First we show that the double exponential size of the interpolants is only due to the repeated occurrence of subformulas and that our algorithm yields single exponential size interpolants using a succinct (DAG-shaped as opposed to tree-shaped) concept representation. Next we use a result of Avigad [Avigad, 2003] showing that succinctly represented first-order formulas can be transformed in polynomial time into equivalent ordinary tree-shaped first-order formulas over structures with at least two elements. This allows us to compute single exponential first-order interpolants over structures with at least two elements. After that, we show that single exponential interpolants over structures with one element can be constructed by a reduction to propositional logic. By combining the interpolants obtained via these two methods, we finally obtain the desired single exponential first-order interpolant over arbitrary structures.

## 4 Results on Beth Definability

In this section, we present the main technical contributions of the paper. We start by a positive result on BP which is a direct application of the interpolation theorem, i.e., Theorem 3.2.

**Theorem 4.1 (BP).** *Let $\mathscr{L}$ be $\mathcal{ALC}$ or any of its extensions with constructors from $\{\mathcal{S}, \mathcal{I}, \mathcal{F}\}$. Then for all $\mathscr{L}$-concepts $C$, all $\mathscr{L}$-TBoxes $\mathcal{T}$, and all signatures $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$, if $C$ is implicitly definable from $\Sigma$ under $\mathcal{T}$ then $C$ is explicitly definable from $\Sigma$ under $\mathcal{T}$, and the explicit definition of $C$ can be computed in time double exponential in $|\mathcal{T}| + |C|$.*

*Proof.* Let $\mathscr{L}$ be one of the DLs stated in the theorem, $C$ be an $\mathscr{L}$-concept, $\mathcal{T}$ be an $\mathscr{L}$-TBox, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{T})$ such that $C$ is implicitly definable from $\Sigma$ under $\mathcal{T}$. We have that $\mathcal{T} \cup \widetilde{\mathcal{T}} \models C \equiv \widetilde{C}$ by Lemma 2.3. Now by Theorem 3.2, there is an interpolant $I$ of $C$ and $\widetilde{C}$ under $\langle \mathcal{T}, \widetilde{\mathcal{T}} \rangle$ that can be computed in time double exponential in $|\mathcal{T}| + |\widetilde{\mathcal{T}}| + |C| + |\widetilde{C}|$. Since it is an interpolant, $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{T}) \cap \mathsf{sig}(\widetilde{C}, \widetilde{\mathcal{T}}) = \Sigma$, and both (a) $\mathcal{T} \cup \widetilde{\mathcal{T}} \models C \sqsubseteq I$ and (b) $\mathcal{T} \cup \widetilde{\mathcal{T}} \models I \sqsubseteq \widetilde{C}$. By (b) and $\mathcal{T} \cup \widetilde{\mathcal{T}} \models \widetilde{C} \sqsubseteq C$, we have $\mathcal{T} \cup \widetilde{\mathcal{T}} \models I \sqsubseteq C$, from which $\mathcal{T} \cup \widetilde{\mathcal{T}} \models C \equiv I$ follows by (a). From the structure of $\widetilde{\mathcal{T}}$, it now straightforwardly follows that $\mathcal{T} \models C \equiv I$.

As for the time needed to compute $I$, observe that $|\mathcal{T}| + |\widetilde{\mathcal{T}}| + |C| + |\widetilde{C}| = 2 \cdot (|\mathcal{T}| + |C|)$. Hence $I$ can be computed in time double exponential in $|\mathcal{T}| + |C|$. □

The proof of Theorem 4.1 uses Theorem 3.2. Similarly, if we use Theorem 3.11 instead, we can show that first-order explicit definitions of implicitly defined concepts can be computed in ExpTime. Note that Theorem 4.1 also establishes a double exponential upper bound on the size explicit definitions in the considered logics. This upper bound is optimal because explicit definitions in $\mathscr{L}$ can be inherently very big.

**Theorem 4.2 (Explicit definition lower bound).** *For every $n \in \mathbb{N}$, there is an $\mathcal{ALC}$-concept $C_n$ and an $\mathcal{ALC}$-TBox $\mathcal{T}_n$ such that $|\mathcal{T}_n|$ and $|C_n|$ are polynomial in $n$, $C_n$ is implicitly definable from some $\Sigma \subseteq \mathsf{sig}(C_n, \mathcal{T}_n)$ under $\mathcal{T}_n$, and the smallest explicit definition of $C_n$ is double exponentially long in $n$.*

*Proof.* Let $A_1, \ldots, A_n$ be concept names and let $R, S$ be role names. For $k \in \{1, \ldots, n\}$,

- let $X_k = \neg A_1 \sqcap \ldots \sqcap \neg A_{k-1} \sqcap A_k$ and
- let $Y_k = A_1 \sqcap \ldots \sqcap A_{k-1} \sqcap \neg A_k$.

Note that $X_k$ and $Y_k$ are just abbreviations, not concept names. Now define the $\mathcal{ALC}$-TBox $\mathcal{T}_n$ as follows.

- $\neg A_1 \sqcap \ldots \sqcap \neg A_n \sqsubseteq \forall R.\bot \sqcap \forall S.\bot$
- $A_1 \sqcup \ldots \sqcup A_n \sqsubseteq \exists R.\top \sqcup \exists S.\top$

- For $k = 1 \dots n$ and $\sigma \in \{R, S\}$,

$$X_k \sqsubseteq \forall\sigma.Y_k \sqcap \prod_{k<l\leq n} ((A_l \sqcap \forall\sigma.A_l) \sqcup (\neg A_l \sqcap \forall\sigma.\neg A_l))$$

Note that $|\mathcal{T}_n|$ is polynomial in $n$ and that $\mathcal{T}_n$ is satisfiable. If $\mathcal{I}$ is a model of $\mathcal{T}_n$, we have for every $s \in \Delta^{\mathcal{I}}$ and every $i \in \{1, \dots, n\}$, either $s \in A_i^{\mathcal{I}}$ or $s \in (\neg A_i)^{\mathcal{I}}$ by the virtue of $\mathcal{I}$ being an interpretation. Therefore every $s \in \Delta^{\mathcal{I}}$ is assigned a unique number between $0, \dots, 2^n - 1$ that is expressed in terms of concept names $A_1, \dots, A_n$. For convenience, we will write e.g., 5 for $A_3 \sqcap \neg A_2 \sqcap A_1$ when $n = 3$.

Now define concepts $C_0 \dots C_{2^n-1}$ as follows.

$$\begin{aligned} C_0 &= \forall R.\bot \sqcap \forall S.\bot \\ C_i &= \exists R.C_{i-1} \sqcup \exists S.C_{i-1} \end{aligned}$$

Intuitively, $C_i$ has the shape of a binary tree (due to role names $R, S$) and the height of the tree is exponential in $i$. This implies $|C_{2^n-1}|$ is double exponential in $n$. Moreover, we have for every $i \in \{0, \dots, 2^n - 1\}$, $\mathcal{T}_n \models i \equiv C_i$. This means $i$ is explicitly (and thus by Proposition 2.7 implicitly) definable from $\{R, S\}$ under $\mathcal{T}_n$. The rest of the proof goes on to show that there is no shorter concept than $C_i$ that is an explicit definition of $i$ from $\{R, S\}$ under $\mathcal{T}_n$. In order to show this, we use the path-set construction from [Lutz, 2006].

In conclusion, $\mathcal{T}_n \models 2^n - 1 \equiv C_{2^n-1}$, $|C_{2^n-1}|$ is double exponential in $n$, and there is no shorter concept $C$ than $C_{2^n-1}$ that is an explicit definition of $2^n - 1$ from $\{R, S\}$ under $\mathcal{T}_n$. Hence the lower bound follows.

Note that by the *role disjunction* constructor which is not present in $\mathcal{ALC}$, one could get away with a single exponential explicit definition. Also note that the lower bound argument works for CBP as well. In fact, one just seeks an explicit definition over $\Sigma = \emptyset$ in this case. □

Note that by Theorems 4.2 and 4.1, we have that implicit definitions using general TBoxes are double exponentially more succinct than acyclic concept definitions. This closes the open problem of [ten Cate *et al.*, 2006] about the size of explicit definitions. Moreover, the same theorems also establish an exact bound on the size of equivalent rewritings of concept queries as considered in [Seylan *et al.*, 2009].

By the following theorem, we have that BP fails in the considered logics having $\mathcal{H}$. This shows that BP is indeed a stronger property than CBP because the same logics have CBP [ten Cate *et al.*, 2006] (cf. Section 2). The theorem also implies that the claim in [Seylan *et al.*, 2010] stating that $\mathcal{ALCH}$ and its extensions with $\mathcal{S}$ and/or $\mathcal{I}$ have BP is incorrect.

**Theorem 4.3.** *Let $\mathscr{L}$ be $\mathcal{ALCH}$ or any of its extensions with constructors from $\{\mathcal{S}, \mathcal{I}, \mathcal{F}\}$. Then $\mathscr{L}$ does not have BP.*

*Proof.* Consider the $\mathcal{ALCH}$-TBox $\mathcal{T}$ which consists of

$$\begin{aligned} S &\sqsubseteq R_1 \\ S &\sqsubseteq R_2 \\ \exists R_1.A \sqcap \forall S.\bot &\sqsubseteq \forall R_2.\neg A \\ \exists R_1.\neg A \sqcap \forall S.\bot &\sqsubseteq \forall R_2.A \end{aligned}$$

It is easy to see that $\mathcal{T}$ is satisfiable. For an interpretation $\mathcal{I}$, define $X_{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \exists t \in \Delta^{\mathcal{I}}.\langle s, t\rangle \in R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}\}$. We have that for all models $\mathcal{I}$ of $\mathcal{T}$, $(\exists S.\top)^{\mathcal{I}} = X_{\mathcal{I}}$. This implies that $\exists S.\top$ is implicitly definable from $\Sigma = \{R_1, R_2\}$ under $\mathcal{T}$.

We now show that there is no $\mathcal{ALCFI}$-concept $C$ such that $\mathsf{sig}(C) \subseteq \Sigma$ and $\mathcal{T} \models \exists S.\top \equiv C$. To this aim, let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be the interpretation where

- $\Delta^{\mathcal{I}} = \{s, t\}$,
- $R_1^{\mathcal{I}} = R_2^{\mathcal{I}} = S^{\mathcal{I}} = \{\langle s, t\rangle\}$;
- $B^{\mathcal{I}} = \emptyset$, for all $B \in N_C$.

Let $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ be the interpretation where

- $\Delta^{\mathcal{J}} = \{w, v, a, b\}$,
- $R_1^{\mathcal{J}} = \{\langle w, a\rangle, \langle v, b\rangle\}$, $R_2^{\mathcal{J}} = \{\langle w, b\rangle, \langle v, a\rangle\}$, $S^{\mathcal{J}} = \emptyset$;
- $A^{\mathcal{J}} = \{a\}$ and for all $B \in (N_C \setminus \{A\})$, $B^{\mathcal{J}} = \emptyset$.

$\mathcal{I}$ and $\mathcal{J}$ are models of $\mathcal{T}$. Using a bisimulation argument, one can show that $s$ and $w$ satisfy the same $\mathcal{ALCFI}$-concepts formulated over $\Sigma$. But clearly, $s \in (\exists S.\top)^{\mathcal{I}}$ and $w \notin (\exists S.\top)^{\mathcal{J}}$.

Now let $\mathscr{L}$ be as stated in the theorem, i.e., it is either $\mathcal{ALCH}$ or any of its extension with constructors from $\{\mathcal{S}, \mathcal{I}, \mathcal{F}\}$. Since the concept language of $\mathcal{SHIF}$ is same as the concept language of $\mathcal{ALCFI}$ and no transitive role occurs in $\mathcal{T}$, we have that $\exists S.\top$ is not explicitly definable from $\Sigma$ under $\mathcal{T}$ in $\mathcal{SHIF}$. But then there is no explicit definition in $\mathscr{L}$. Hence $\mathscr{L}$ does not have BP. □

As discussed in Section 2, it also makes sense to study BPF. However in the considered logics lacking FMP, BPF fails.

**Theorem 4.4.** *Let $\mathscr{L}$ be $\mathcal{ALCFI}$ or any of its extensions with constructors from $\{\mathcal{S}, \mathcal{H}\}$. Then $\mathscr{L}$ does not have BPF.*

*Proof.* Let $A, B, X$ be concept names and let $R$ be a role name. Consider the $\mathcal{ALCFI}$-TBox $\mathcal{T}$ which consists of the following.

$$\begin{aligned} \top &\sqsubseteq \leq 1R^- \\ B &\sqsubseteq \exists R.B \\ A &\sqsubseteq X \\ \exists R.(A \sqcap \neg B) &\sqsubseteq \neg X \\ \exists R.\neg X &\sqsubseteq \neg X \end{aligned}$$

It is easy to see that the concept $A \sqcap B$ is finitely satisfiable w.r.t. $\mathcal{T}$, i.e., there is some finite model $\mathcal{I}$ of $\mathcal{T}$ such that $(A \sqcap B)^{\mathcal{I}} \neq \emptyset$. In fact, we provide two such models $\mathcal{I}_n$ and $\mathcal{J}_n$ later on in the proof.

Now for all interpretations, define

$$Y_{\mathcal{I}} = \{s \in \Delta^{\mathcal{I}} \mid \exists t \in \Delta^{\mathcal{I}}.\langle s, t\rangle \in (R^{\mathcal{I}})^+ \wedge t \in A^{\mathcal{I}}\} \cap A^{\mathcal{I}}$$

where $(R^{\mathcal{I}})^+$ is the transitive closure of $R^{\mathcal{I}}$. We claim that for all finite models $\mathcal{I}$ of $\mathcal{T}$, we have $(A \sqcap B)^{\mathcal{I}} = Y_{\mathcal{I}}$. In particular, this implies $A \sqcap B$ is f-implicitly definable from $\Sigma = \{R, A\}$ under $\mathcal{T}$.

For some $n \in \mathbb{N}$, we will define in the following two interpretations $\mathcal{I}_n, \mathcal{J}_n$. Let $s_0, \dots, s_{2n+1}$ be all distinct elements

and define $\mathcal{I}_n = \langle \{s_0, \ldots, s_{2n+1}\}, \cdot^{\mathcal{I}_n} \rangle$ to be the interpretation which consists of the $R$-cycle[1] $s_0, s_1, \ldots, s_{2n+1}, s_0$, where $A^{\mathcal{I}_n} = \{s_0\}$, $X^{\mathcal{I}_n} = \{s_0\}$, and $B^{\mathcal{I}_n} = \Delta^{\mathcal{I}_n}$; and define $\mathcal{J}_n = \langle \{s_0, \ldots, s_{2n+1}\} \setminus \{s_{n+1}\}, \cdot^{\mathcal{J}_n} \rangle$ to be the interpretation which consists of the $R$-path

$$s_{n+2}, s_{n+3}, \ldots, s_{2n+1}, s_0, s_1, \ldots, s_n$$

where $A^{\mathcal{J}_n} = \{s_0\}$, $X^{\mathcal{J}_n} = \{s_0\}$, and $B^{\mathcal{J}_n} = \emptyset$.

Observe that $\mathcal{I}_n$ and $\mathcal{J}_n$ are (finite) models of $\mathcal{T}$, and $\mathcal{J}_n$ is the subinstance of $\mathcal{I}_n$ where the element $s_{n+1}$ is removed.

**Claim 4.5.** *For every $\mathcal{ALCFI}$-concept $C$ such that $\mathsf{sig}(C) \subseteq \Sigma = \{R, A\}$ and $\mathsf{md}(C) \leq n$, where $\mathsf{md}(C)$ is the modal depth of $C$, it holds that $s_0 \in C^{\mathcal{I}}$ iff $s_0 \in C^{\mathcal{J}}$.*

The claim follows from the fact that the subinstances of $\mathcal{I}_n$ and $\mathcal{J}_n$ containing all nodes at distance at most $n$ from $s_0$ are isomorphic. Now we have that $s_0 \in (A \sqcap B)^{\mathcal{I}}$ and $s_0 \notin (A \sqcap B)^{\mathcal{J}}$. But by the previous claim, $s_0$ in $\mathcal{I}$ satisfies exactly the same concepts as $s_0$ in $\mathcal{J}$ that are formulated over $\Sigma$ and with modal depth $\leq n$. Since we can come up with such witness models for every $n$, there exists no $\mathcal{ALCFI}$-concept $C$ such that $\mathsf{sig}(C) \subseteq \Sigma$ and $\mathcal{T} \models_f A \sqcap B \equiv C$. This means $A \sqcap B$ is not f-explicitly definable from $\Sigma$ under $\mathcal{T}$. Hence $\mathcal{ALCFI}$ does not have BPF.

Let $\mathscr{L}$ be any proper extension of $\mathcal{ALCFI}$ with constructors from $\{\mathcal{S}, \mathcal{H}\}$. Since $\mathcal{T}$ and $A \sqcap B$ are respectively an $\mathscr{L}$-TBox and an $\mathscr{L}$-concept, and the concept languages of $\mathcal{ALCFI}$ and $\mathscr{L}$ are the same, we have that $\mathscr{L}$ does not have BPF. □

## 5 Discussion

Qualified number restrictions, denoted by $\mathcal{Q}$ in the language, is a generalization of $\mathcal{F}$. Extending our upper bound results on the size of explicit definitions to $\mathcal{ALCQ}$ and $\mathcal{ALCQI}$ appears to be difficult. This is because of the unavailability of a natural and optimal tableau algorithm for these logics. We leave as another open problem the lower bound on the size of the first-order explicit definition given that a concept is implicitly defined under a TBox.

## References


[Avigad, 2003] Jeremy Avigad. Eliminating definitions and skolem functions in first-order logic. *ACM Trans. Comput. Logic*, 4:402–415, July 2003.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003.

[Beth, 1953] E. W. Beth. On Padoa's methods in the theory of definitions. *Indagationes Mathematicae*, 15:330–339, 1953.

[Calvanese *et al.*, 2001] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In *Handbook of Automated Reasoning*, pages 1581–1634, 2001.

[Craig, 1957] William Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, 22(3):269–285, 1957.

[Duc and Lamolle, 2010] Chan Le Duc and Myriam Lamolle. Decidability of description logics with transitive closure of roles in concept and role inclusion axioms. In *Description Logics*, 2010.

[Fitting, 1996] Melvin Fitting. *First-order logic and automated theorem proving (2nd ed.)*. Springer-Verlag, 1996.

[Gabbay and Maksimova, 2005] Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability in Modal Logics (Oxford Logic Guides)*. Clarendon Press, 2005.

[Goranko and Otto, 2007] Valentin Goranko and Martin Otto. Model theory of modal logic. In *Handbook of Modal Logic*, pages 249 – 329. Elsevier, 2007.

[Goré and Nguyen, 2007] Rajeev Goré and Linh Anh Nguyen. Exptime tableaux for $\mathcal{ALC}$ using sound global caching. In *Description Logics*, 2007.

[Hoogland, 2001] Eva Hoogland. *Definability and Interpolation: Model-theoretic investigations*. PhD thesis, University of Amsterdam, 2001.

[Horrocks *et al.*, 2000] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[Horrocks *et al.*, 2003] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[Lang and Marquis, 2008] Jérôme Lang and Pierre Marquis. On propositional definability. *Artif. Intell.*, 172:991–1017, May 2008.

[Lutz *et al.*, 2005] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *J. Artif. Intell. Res. (JAIR)*, 23:667–726, 2005.

[Lutz, 2006] Carsten Lutz. Complexity and succinctness of public announcement logic. In *AAMAS*, pages 137–143, 2006.

[Nash *et al.*, 2010] Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, 35(3), 2010.

[Seylan *et al.*, 2009] Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over DBoxes. In *IJCAI*, pages 923–929, 2009.

[Seylan *et al.*, 2010] Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Optimal rewritings in definitorially complete description logics. In *Description Logics*, 2010.

[ten Cate *et al.*, 2006] Balder ten Cate, Willem Conradie, Maarten Marx, and Yde Venema. Definitorially complete description logics. In *KR*, pages 79–89, 2006.

[Tobies, 2001] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH-Aachen, 2001.


[1] an $R$-path such that the start and the end nodes are the same