Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

Optimal Rewritings in Definitorially Complete Description Logics*

İnanç Seylan, Enrico Franconi, and Jos de Bruijn

Free University of Bozen-Bolzano, Italy
{seylan,franconi,debruijn}@inf.unibz.it

Abstract. In this paper, we revisit the problem of *definitorial completeness*, i.e., whether a given general TBox $\mathcal{T}$ in a description logic (DL) $\mathscr{L}$ can be rewritten to an acyclic TBox $\mathcal{T}'$ in $\mathscr{L}$. This is an important problem because crucial optimisations in DL reasoners rely on acyclic parts in TBoxes. It is known that such rewritings are possible for *definitorial* TBoxes in $\mathcal{ALC}$ and in logics $\mathcal{ALC}X$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$. Here we establish optimal bounds on the sizes of the resulting acyclic TBoxes. In particular, we reduce the known triple exponential upper bound on $\mathcal{ALC}$-TBoxes to single exponential. Additionally, we prove the same upper bound for those extensions with $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ for which there was no established result before. This means, together with the already known exponential lower bound for $\mathcal{ALC}$, that our bounds are tight.

1 Introduction

Description logic (DL) TBoxes enable one to introduce names for complex concepts using *concept definitions*. For example, the definition $Parent \equiv Mother \sqcup Father$ classifies all individuals that are either mothers or fathers as parents. Here, $Parent$ is called a *defined* concept, and $Mother$ and $Father$ are *primitive* concepts. In some sense, instances of primitive concepts come directly from the application domain whereas defined concepts help us to define views or constraints. Baader and Nutt [1] call a finite set of concept definitions a *terminology* if no concept name is defined more than once.

Terminologies can be cyclic, i.e., a defined concept may refer to itself directly in its definition or indirectly through some other defined concept. Cyclicity is a syntactic condition and for certain cyclic terminologies there may be equivalent acyclic ones. For example, the definition

$$Parent \equiv (Parent \sqcup \neg Parent) \sqcap (Mother \sqcup Father)$$

contains the tautological expression $(Parent \sqcup \neg Parent)$. By removing this expression we obtain an equivalent acyclic definition.

Acyclic TBoxes are of particular interest because reasoning with them is "easier" than with general TBoxes. For example, satisfiability of an acyclic $\mathcal{ALC}$-TBox is a PSPACE-complete problem whereas the variant of the problem for general $\mathcal{ALC}$-TBoxes is EXPTIME-complete [2]. On the practical side of things, absorption is an

* The first author would like to thank Maarten Marx for his hospitality and the stimulating discussions. This work has been partially supported by the EU project Ontorule.

indispensable optimisation technique in DL reasoners which makes use of the acyclic part of a TBox [3]. Therefore a natural question arises: from which cyclic terminologies can we obtain equivalent acyclic ones? Baader and Nutt answer this question by identifying a semantic condition on terminologies called *definitoriality* [1]. Intuitively, if a terminology is definitorial and the instances of primitive concepts are known then the instances of defined concepts are completely determined. In particular, Baader and Nutt show that $\mathcal{ALC}$ is *definitorially complete*, i.e., for every definitorial $\mathcal{ALC}$-terminology there is an equivalent acyclic $\mathcal{ALC}$-terminology. As also noted by the authors, definitorial completeness is a form of Beth Definability [4] – a property of first-order logic – for DLs.

Another relevant question which is of practical interest is how an equivalent acyclic terminology can be obtained from a definitional one. Ten Cate et al. [5] give a constructive method for calculating an acyclic $\mathcal{ALC}$-terminology from a definitorial one and prove definitorial completeness for some extensions of $\mathcal{ALC}$. To be more precise, Ten Cate et al. consider the same problem for general TBoxes instead of terminologies. In general TBoxes, it is not clear from the syntactic shape of the TBox anymore which predicate is primitive and which is defined. In this setting, primitive predicates are assumed as given. Moreover, Ten Cate et al. establish a single exponential lower and a triple exponential upper bound on the size of the generated TBoxes in $\mathcal{ALC}$. However, the exact characterisation of the succinctness of general TBoxes over acyclic ones was left as an open problem.

In this paper, we reduce the upper bound on the size of the equivalent acyclic terminologies obtained from definitorial $\mathcal{ALC}$-TBoxes to single exponential, which is tight. We then extend this result to all logics $\mathcal{ALC}X$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$, for which there were no earlier established results. In previous work [6], we used Beth Definability (adapted to DLs) to rewrite a given concept into an equivalent one for efficient instance retrieval using databases. Our results in this paper extend to that scenario as well. More precisely, here we give an optimal version of the algorithm that computes rewritings.

We start by giving a brief introduction to standard notions we will use from DLs in Section 2. In Section 3 we give our main result for $\mathcal{ALC}$ after introducing relevant terminology. These results are based on the algorithm we present in Section 4. Our results for extensions of $\mathcal{ALC}$ are presented in Section 5, after which we conclude.

2 Preliminaries

Let N_C and N_R be countably infinite and disjoint sets of *concept* and *role* names, respectively. With N_P we denote the set of *predicates* $N_C \cup N_R$.

The set of *$\mathcal{SHI}$-roles* is defined as $N_R \cup \{R^- \mid R \in N_R\}$. A *role inclusion axiom* is of the form $R \sqsubseteq S$, with R and S $\mathcal{SHI}$-roles. A *transitivity axiom* is of the form $\mathsf{Trans}(R)$, for R a $\mathcal{SHI}$-role. A *role hierarchy* $\mathcal{H}$ is a finite set of role inclusion and transitivity axioms.

For a role hierarchy $\mathcal{H}$, we define the function Inv over roles as $\mathsf{Inv}(R) := R^-$ if $R \in N_R$ and $\mathsf{Inv}(R) := S$ if $R = S^-$, for some $S \in N_R$. Further we define $\sqsubseteq_{\mathcal{H}}$ as the smallest transitive reflexive relation on $\mathcal{SHI}$-roles in $\mathcal{H}$ such that $R \sqsubseteq S \in \mathcal{H}$ implies $R \sqsubseteq_{\mathcal{H}} S$ and $\mathsf{Inv}(R) \sqsubseteq_{\mathcal{H}} \mathsf{Inv}(S)$.

The set of $\mathcal{SHI}$*-concepts* and their semantics are defined in the standard way [7]. A $\mathcal{SHI}$*-TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$ and/or *concept definitions* $A \equiv C$, where A is a concept name, and C and D are $\mathcal{SHI}$-concepts. A $\mathcal{SHI}$ *knowledge base* (KB) $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{H})$, where $\mathcal{T}$ is a $\mathcal{SHI}$-TBox and $\mathcal{H}$ is a role hierarchy. For a $\mathcal{SHI}$-concept C and a $\mathcal{SHI}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{H})$, $\mathsf{rol}(C, \mathcal{K})$ and $\mathsf{sig}(C, \mathcal{K})$ denote, respectively, the sets of role and predicate (i.e., concept or role) names occurring in C or $\mathcal{K}$. We are interested in special acyclic TBoxes.

Definition 1 ([5]). *Let $\mathcal{T}$ be a TBox. A concept name A* directly uses *a concept name B in $\mathcal{T}$ if there is some $A \equiv C \in \mathcal{T}$ and $B \in \mathsf{sig}(C)$;* uses *is the transitive closure of the relation* directly uses*.*

Let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is Σ-acyclic *if it satisfies the following two properties:*

1. *$\mathcal{T}$ consists of exactly one concept definition $A \equiv C$ for each concept name $A \in (\mathsf{sig}(\mathcal{T}) \setminus \Sigma)$, plus a number of concept inclusion axioms $C \sqsubseteq D$, where $\mathsf{sig}(C) \subseteq \Sigma$ and $\mathsf{sig}(D) \subseteq \Sigma$.*
2. *There is no concept name A that uses itself in $\mathcal{T}$.*

The notion of an interpretation satisfying a role hierarchy or TBox is defined in the usual way (cf. [7]). An interpretation $\mathcal{I}$ satisfies $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ if and only if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{H}$. In this case, we say that $\mathcal{I}$ is a model of $\mathcal{K}$. $\mathcal{K}$ is *satisfiable* if $\mathcal{K}$ has a model. Two KBs are *equivalent* if they have the same models. A concept C is *satisfiable w.r.t.* $\mathcal{K}$ if and only if there is some model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. The concept subsumption and equivalence problems, i.e., checking whether $\mathcal{K} \models C \sqsubseteq D$ (respectively, $(\mathcal{K} \models C \equiv D)$, are defined in the usual way.

A concept C is in *negation normal form* (NNF) if and only if the negation sign appears only in front of concept names in C. A concept can be transformed into an equivalent one in NNF in linear time and thus, we assume all concepts to be in NNF. For a concept C, we denote its negation in NNF by $\dot{\neg}C$. Moreover, we will sometimes consider only concept inclusions of the form $\top \sqsubseteq C$ to which every concept inclusion and definition can be rewritten again in linear time.

The *concept closure* $\mathsf{cl}(C_0, \mathcal{K})$ of C_0 and $\mathcal{K}$ is the smallest set of concepts satisfying the following conditions:

- $C_0 \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $\top \sqsubseteq C \in \mathcal{T}$ then $C \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $C \in \mathsf{cl}(C_0, \mathcal{K})$ and D is a subconcept of C then $D \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $\forall R.C \in \mathsf{cl}(C_0, \mathcal{K})$, $S \sqsubseteq_{\mathcal{H}} R$, and $\mathsf{Trans}(S) \in \mathcal{H}$ then $\forall S.C \in \mathsf{cl}(C_0, \mathcal{K})$.

We define the notions of closure $\mathsf{f}(e)$, for $\mathsf{f} \in \{\mathsf{sig}, \mathsf{cl}, \mathsf{rol}\}$ and $e \in \{C, \mathcal{T}, \mathcal{H}, \mathcal{K}\}$, analogously. The *size* of a concept C (written $|C|$) is the number of elements in $\mathsf{cl}(C)$. For a TBox $\mathcal{T}$, $|\mathcal{T}| := \sum_{\top \sqsubseteq C \in \mathcal{T}} |C|$.

3 Beth Definability

We introduce in this Section implicit and explicit *definability* for concepts. We used these notions in [6] to reduce the instance retrieval problem in DLs with DBoxes to

SQL query answering. In this section, we will use them again to rewrite definitorial TBoxes into acyclic ones in a more direct way than Ten Cate et al. did in [5]. We start by giving a semantic characterisation of implicit definability.

Definition 2 (Reduct). *Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be an interpretation and let $\Sigma \subseteq N_P$. An interpretation $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ is the* reduct *of $\mathcal{I}$ to Σ (denoted by $\mathcal{I}|_\Sigma$) if and only $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $\cdot^{\mathcal{J}}$ is defined only on the symbols in Σ.*

Definition 3 (Implicit definability). *Let C be a concept, $\mathcal{K}$ a KB, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. C is* implicitly definable *from Σ under $\mathcal{K}$ if and only if for any two models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{K}$, $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$ implies $C^{\mathcal{I}} = C^{\mathcal{J}}$.*

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 Σ.

Example 1. Consider the KB $\mathcal{K} = (\mathcal{T}, \emptyset)$, where $\mathcal{T}$ is equal to:

$$\begin{aligned} \mathit{Project} &\sqsubseteq \mathit{Activity} \\ \mathit{Meeting} &\sqsubseteq \mathit{Activity} \\ \mathit{Activity} &\sqsubseteq \mathit{Project} \sqcup \mathit{Meeting} \\ \mathit{Project} &\sqsubseteq \neg \mathit{Meeting} \end{aligned}$$

and let $\Sigma = \{\mathit{Meeting}, \mathit{Activity}\}$. $\mathit{Project}$ is implicitly definable from Σ under $\mathcal{K}$ since its extension depends only on the (fixed) extension of $\mathit{Meeting}$ and $\mathit{Activity}$.

The following proposition provides an alternative, syntactic definition of implicit definability. In particular, it reduces checking implicit definability to the entailment problem in the same logic. Let a concept $\widetilde{C}$ (resp., KB $\widetilde{\mathcal{K}}$) be like C (resp., $\mathcal{K}$) except that every occurrence of each predicate $P \in (\Sigma \setminus \mathsf{sig}(C))$ (resp. $P \in (\Sigma \setminus \mathsf{sig}(\mathcal{K}))$) is replaced with a new predicate $\widetilde{P}$.

Proposition 1. *A concept C is implicitly definable from Σ under $\mathcal{K}$ if and only if $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$.*

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

Definition 4 (Explicit definability). *Let C be a concept, $\mathcal{K}$ a KB, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. C is* explicitly definable *from Σ under $\mathcal{K}$ if and only if there is some concept D such that $\mathcal{K} \models C \equiv D$ and $\mathsf{sig}(D) \subseteq \Sigma$. Such a D is called an* explicit definition *of C from Σ under $\mathcal{K}$.*

In Example 1, the explicit definition of $\mathit{Project}$ is $\mathit{Activity} \sqcap \neg \mathit{Meeting}$. It is not hard to see that explicit definability implies implicit definability. Beth [4] shows that the converse holds for the case of first-order logic: if C is implicitly definable from Σ in $\mathcal{K}$, then it is explicitly definable. This property for $\mathcal{ALC}$ with general TBoxes is proved in [6] by exploiting interpolation. Here we state a stronger version of the theorem in [6] by putting an exponential bound on the size of the explicit definition and give the proof again to show how interpolation is used.

Definition 5. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be a KB. A* labelling *of $\mathcal{K}$ is an ordered pair $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ of KBs where $\mathcal{K}_\mathbf{l} = (\mathcal{T}_\mathbf{l}, \mathcal{H}_\mathbf{l})$, $\mathcal{K}_\mathbf{r} = (\mathcal{T}_\mathbf{r}, \mathcal{H}_\mathbf{r})$, $\mathcal{T} = \mathcal{T}_\mathbf{l} \cup \mathcal{T}_\mathbf{r}$, and $\mathcal{H} = \mathcal{H}_\mathbf{l} \cup \mathcal{H}_\mathbf{r}$; $\langle \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ is a labelling of the TBox $\mathcal{T}$.*

Definition 6 (Interpolant). *Let C, D be concepts and let $\mathcal{K}$ be a KB such that $\mathcal{K} \models C \sqsubseteq D$. A concept I is called an* interpolant *of C and D under a labelling $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ of $\mathcal{K}$ if $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{K}_\mathbf{l}) \cap \mathsf{sig}(D, \mathcal{K}_\mathbf{r})$, $\mathcal{K} \models C \sqsubseteq I$, and $\mathcal{K} \models I \sqsubseteq D$.*

Section 4 is devoted to a constructive proof for the following lemma by using an optimal tableau calculus for $\mathcal{ALC}$.

Lemma 1. *Let C and D be $\mathcal{ALC}$-concepts and let $\mathcal{K} = \mathcal{T}$ be an $\mathcal{ALC}$-KB such that $\mathcal{K} \models C \sqsubseteq D$. If $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ is a labelling of $\mathcal{K}$ then there exists an interpolant of C and D under $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ whose size is at most exponential in $|\mathcal{T}| + |C| + |D|$.*

Theorem 1 (Beth Definability). *Let C be an $\mathcal{ALC}$-concept, let $\mathcal{K} = \mathcal{T}$ be an $\mathcal{ALC}$-KB, and let $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. If C is implicitly definable from Σ under $\mathcal{K}$ then C is explicitly definable from Σ under $\mathcal{K}$, and the size of the explicit definition is at most exponential in $|\mathcal{T}| + |C|$.*

Proof. We have that $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$ by implicit definability of C. Moreover, $\langle \mathcal{K}, \widetilde{\mathcal{K}} \rangle$ is a labelling of $\mathcal{K} \cup \widetilde{\mathcal{K}}$. Now, by Lemma 1 and $|C| = |\widetilde{C}|$, there is an interpolant I of C and $\widetilde{C}$ under $\langle \mathcal{K}, \widetilde{\mathcal{K}} \rangle$ and the size of I is at most exponential in $|\mathcal{T}| + |C|$. Since it is an interpolant, $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{K}) \cap \mathsf{sig}(\widetilde{C}, \widetilde{\mathcal{K}}) = \Sigma$, and both (a) $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \sqsubseteq I$ and (b) $\mathcal{K} \cup \widetilde{\mathcal{K}} \models I \sqsubseteq \widetilde{C}$. By (b) and $\mathcal{K} \cup \widetilde{\mathcal{K}} \models \widetilde{C} \sqsubseteq C$, we have $\mathcal{K} \cup \widetilde{\mathcal{K}} \models I \sqsubseteq C$, from which $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv I$ follows by (a). From the structure of $\widetilde{\mathcal{K}}$ and the fact that $\mathsf{sig}(C), \mathsf{sig}(I) \subseteq \mathsf{sig}(\mathcal{K})$ straightforwardly follows that $\mathcal{K} \models C \equiv I$. □

This proof of Beth definability for $\mathcal{ALC}$ with general TBoxes is constructive, provided we have a constructive method of finding interpolants as defined in Definition 6. As we will see in Section 4, this constructive method is based on tableau. To be more precise, the tableau algorithm will allow us to check whether a concept is implicitly definable and if this is the case, we will use the same tableau proof to construct an explicit definition. Note that Theorem 1 also establishes a single exponential upper bound on the size of explicit definitions we calculate. Together with the following theorem which establishes the lower bound, we can conclude that our procedure for calculating explicit definitions is worst-case optimal.

Theorem 2 ([5]). *There are an $\mathcal{ALC}$-concept C, $\mathcal{ALC}$-KB $\mathcal{K} = \mathcal{T}$, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$ such that C is implicitly definable from Σ under $\mathcal{K}$ and the smallest explicit definition of C is exponential in $|C| + |\mathcal{K}|$.*

We now formally define the notions we discussed in the introduction. However, unlike Baader and Nutt [1], we consider general TBoxes instead of terminologies. In general TBoxes, it is not clear from the syntactic shape of the TBox which predicate is primitive and which is defined. Therefore, we assume that primitive predicates, i.e., Σ, are specified beforehand. This is similar to the approach by Ten Cate et al. [5].

Definition 7. *Let $\mathcal{T}$ be a TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is* Σ-definitorial *if and only if for every interpretation $\mathcal{I}$ that interprets only the predicates in Σ there is at most one interpretation $\mathcal{J}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, $P^{\mathcal{I}} = P^{\mathcal{J}}$ for every predicate $P \in \Sigma$, and $\mathcal{J}$ is a model of $\mathcal{T}$.*

It is not hard to see the connection between definitoriality and implicit definability.

Theorem 3. *Let $\mathcal{T}$ be a TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is Σ-definitorial if and only if every concept name $A \in \mathsf{sig}(\mathcal{T}) \setminus \Sigma$ is implicitly definable from Σ under $\mathcal{T}$.*

We are interested in rewriting general TBoxes to Σ-acyclic ones. It is clear from the definition of Σ-acyclic TBoxes that they may contain general concept inclusions involving only predicates from Σ. This restriction is needed because unlike in [1] we may be given a TBox that is not a terminology. A Σ-acyclic TBox is also Σ-definitorial, but the converse may not always be true. DLs possessing this property are called definitorially complete.

Definition 8. *A description logic $\mathscr{L}$ is called* definitorially complete *if each Σ-definitorial $\mathscr{L}$-TBox $\mathcal{T}$ is equivalent to a Σ-acyclic $\mathscr{L}$-TBox $\mathcal{T}'$.*

Baader and Nutt show that $\mathcal{ALC}$ is definitorially complete [1]. Ten Cate et al. give a concrete algorithm for computing acyclic TBoxes from definitorial ones in $\mathcal{ALC}$ [5]. The algorithm is based on a special normal form for concepts and uniform interpolation. This involves at most a triple exponential blowup. Here we take a more direct approach using interpolation and improve this upper bound to a single exponential one, which is the main result of this section.

Theorem 4. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. If $\mathcal{T}$ is Σ-definitorial, then there exists an equivalent Σ-acyclic $\mathcal{ALC}$-TBox $\mathcal{T}^*$, which is at most exponential in the size of $\mathcal{T}$.*

Proof. Let $\mathcal{T}$ be a Σ-definitorial $\mathcal{ALC}$-TBox. By Theorem 3 and Theorem 1, for every $A \in (\mathsf{sig}(\mathcal{T}) \setminus \Sigma)$, there is some concept C_A such that $\mathcal{T} \models A \equiv C_A$, $\mathsf{sig}(C_A) \subseteq \Sigma$, and $|C_A|$ is at most exponential in $|A| + |\mathcal{T}|$. However, since $A \in \mathsf{sig}(\mathcal{T})$, we can conclude that $|C_A|$ is at most exponential only in $|\mathcal{T}|$.

Let $\mathcal{T}^*$ be the TBox obtained from $\mathcal{T}$ by systematically replacing each occurrence of all A by C_A, and adding the relevant concept definitions $A \equiv C_A$. Then $\mathcal{T}^*$ is Σ-acyclic and equivalent to $\mathcal{T}$. Finally, the length of $\mathcal{T}^*$ is easily seen to be at most exponential in the length of $\mathcal{T}$. □

4 Optimally Constructing Interpolants

In this section, we give a constructive proof of Lemma 1. In other words, we present a method for constructing an interpolant using a tableau proof. We have presented a constructive method in [6]. However, the algorithm there is based on standard $\mathcal{ALC}$ tableau techniques, which do not guarantee termination in ExpTime, and are not worst-case optimal, since checking satisfiability in $\mathcal{ALC}$ is known to be in ExpTime. Here we aim at obtaining exponential size interpolants by using a worst-case optimal tableau algorithm in the style of Goré and Nyugen [8].

The $\mathsf{R}_\bot$ rule	
Condition:	$\{C^\lambda, (\dot{\neg}C)^\kappa\} \subseteq g.\mathsf{content}$.
Action:	$g.\mathsf{status} := \texttt{unsat}$.
The $\mathsf{R}_\sqcap$ rule	
Condition:	$(C_1 \sqcap C_2)^\lambda \in g.\mathsf{content}$, $\{C_1^\lambda, C_2^\lambda\} \not\subseteq g.\mathsf{content}$.
Action:	$g'.\mathsf{content} := g.\mathsf{content} \cup \{C_1^\lambda, C_2^\lambda\}$;
The $\mathsf{R}_\sqcup$ rule	
Condition:	$(C_1 \sqcup C_2)^\lambda \in g.\mathsf{content}$, $\{C_1^\lambda, C_2^\lambda\} \cap g.\mathsf{content} = \emptyset$.
Action:	$g'.\mathsf{content} := g.\mathsf{content} \cup \{C_1^\lambda\}$;
	$g''.\mathsf{content} := g.\mathsf{content} \cup \{C_2^\lambda\}$;
The $\mathsf{R}_\exists$ rule	
Condition:	$\{(\exists R_1.C_1)^{\lambda_1}, \ldots, (\exists R_n.C_n)^{\lambda_n}\} \subseteq g.\mathsf{content}$;
	$(\exists R.C)^\lambda \in g.\mathsf{content}$ implies $\exists i \in \{1, \ldots, n\}$ s.t. $(\exists R.C)^\lambda = (\exists R_i.C_i)^{\lambda_i}$.
Action:	$g_i.\mathsf{content} := \{C_i^{\lambda_i}\} \cup \{D^\lambda \mid (\forall R.D)^\lambda \in g.\mathsf{content}$ and $R_i = R\}$,
	$g_i.\mathsf{content} := g_i.\mathsf{content} \cup \{E^{\mathbf{l}} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{l}\} \cup \{E^{\mathbf{r}} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{r}\}$
	for $1 \leq i \leq n$.

Fig. 1. Tableau expansion rules for $\mathcal{ALC}$.

4.1 An Optimal Tableau Algorithm for Satisfiability

We start by presenting a tableau algorithm for deciding concept subsumption. To this aim, we fix two $\mathcal{ALC}$-concepts C and D, and an $\mathcal{ALC}$-TBox $\mathcal{T}$ with the labelling $\langle \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$. A *biased tableau* (tableau for short) for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ 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.

In the following, we will be using *biased concepts* which are expressions of the form C^λ, where C is an $\mathcal{ALC}$-concept and $\lambda \in \{\mathbf{l}, \mathbf{r}\}$ is a bias. Let $\mathsf{cll} := \{E^\mathbf{l} \mid E \in \mathsf{cl}(C, \mathcal{T}_\mathbf{l})\}$ and $\mathsf{clr} := \{E^\mathbf{r} \mid E \in \mathsf{cl}(\dot{\neg}D, \mathcal{T}_\mathbf{r})\}$. We associate four different labels to nodes in $\mathcal{V}$: $\mathsf{content} : V \to 2^{\mathsf{cll} \cup \mathsf{clr}}$, $\mathsf{type} : V \to \{\texttt{and-node}, \texttt{or-node}\}$, $\mathsf{status} : V \to \{\texttt{sat}, \texttt{unsat}\}$, and $\mathsf{availability} : V \to \{\texttt{expanded}, \texttt{unexpanded}\}$. The function of these labels are explained when they are used.

The *tableau expansion rules* given in Figure 1 expand a tableau by making use of the semantics of concepts, and thus make implicit information explicit. We assume that a rule can be applied to a node g if $g.\mathsf{availability} = \texttt{unexpanded}$ and if a rule is applied to g then $g.\mathsf{availability} := \texttt{expanded}$ without writing it explicitly in rule definitions. 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 $\mathcal{T} \models C \sqsubseteq D$. The tableau algorithm starts with the *initial tableau* $\mathbf{T} = \langle \{g_0\}, \emptyset \rangle$ for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$, where $g_0.\mathsf{content} = \{C^\mathbf{l}, (\dot{\neg}D)^\mathbf{r}\} \cup \{E^\mathbf{l} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{l}\} \cup \{E^\mathbf{r} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{r}\}$ and $g_0.\mathsf{availability} = \texttt{unexpanded}$. $\mathbf{T}$ is then expanded by repeatedly applying the tableau expansion rules in such a way that if more than one rule is applicable at the same time then the first applicable rule in the list $[\mathsf{R}_\bot, \mathsf{R}_\sqcap, \mathsf{R}_\sqcup, \mathsf{R}_\exists]$ is chosen. The expansion continues until none of the rules is applicable to $\mathbf{T}$. Such a tableau is called *complete*.

Let $\mathbf{T}$ be a complete tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$. The type of a node g is determined as follows: $g.\mathsf{type} = \texttt{or-node}$ if $\mathsf{R}_\sqcup$ has been applied to g, and $g.\mathsf{type} = \texttt{and-node}$ otherwise. Until it is no more possible to assign a *status* to a node in $\mathcal{V}$, we run the following algorithm.

- Pick a node $g \in V$.
- If g is a sink node[1] with $g.\mathsf{status} \neq \texttt{unsat}$ then $g.\mathsf{status} := \texttt{sat}$.
- If $g.\mathsf{type} = \texttt{and-node}$ and
  - all g's direct successors have status $\texttt{sat}$ then $g.\mathsf{status} := \texttt{sat}$;
  - one of g's direct successors has status $\texttt{unsat}$ then $g.\mathsf{status} := \texttt{unsat}$.
- If $g.\mathsf{type} = \texttt{or-node}$ and
  - all g's direct successors have status $\texttt{unsat}$ then $g.\mathsf{status} := \texttt{unsat}$;
  - one of g's direct successors has status $\texttt{sat}$ then $g.\mathsf{status} := \texttt{sat}$.

If $g_0.\mathsf{status}$ is still undefined then for every $g \in \mathcal{V}$ with $g.\mathsf{status} \neq \texttt{unsat}$, set $g.\mathsf{status} := \texttt{sat}$.

A complete tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$ is *closed* if g_0 has status $\texttt{unsat}$ and it is *open*, otherwise. If the tableau algorithm constructs an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$ then it returns "$\mathcal{T} \not\models C \sqsubseteq D$", and "$\mathcal{T} \models C \sqsubseteq D$" otherwise.

Termination is a consequence of using proxies and $\mathsf{cl}_\mathbf{l} \cup \mathsf{cl}_\mathbf{r}$ being finite. In the worst case, there are $2^{O(\sharp(\mathsf{cl}_\mathbf{l} \cup \mathsf{cl}_\mathbf{r}))}$ nodes in a complete tableau $\mathbf{T}$. Checking for proxies and determining the status of g_0 both take polynomial number of steps in the size of $\mathbf{T}$. As it is apparent, we use a refutation proof for $\mathcal{T} \models C \sqsubseteq D$, i.e., we check the unsatisfiability of $C \sqcap \dot{\neg} D$ w.r.t. $\mathcal{T}$. For soundness, given a model $\mathcal{I}$ of $\mathcal{T}$ such that $(C \sqcap \dot{\neg} D)^\mathcal{I} \neq \emptyset$, we can guide the tableau algorithm to construct an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$ by making use of the information in $\mathcal{I}$. As for completeness, we can construct a model $\mathcal{I}$ of $\mathcal{T}$ such that $(C \sqcap \dot{\neg} D)^\mathcal{I} \neq \emptyset$ from an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$. Combining all these, we get the following theorem.

Theorem 5 ([8]). *Let C, D be $\mathcal{ALC}$-concepts and $\mathcal{T}$ be an $\mathcal{ALC}$-TBox. The tableau algorithm decides $\mathcal{T} \models C \sqsubseteq D$ in time exponential in $|C| + |D| + |\mathcal{T}|$.*

4.2 An Algorithm for Calculating Interpolants

As is clear from the definition of the tableau expansion rules, we use some additional bookkeeping (compared with the algorithm of [8]) for calculating interpolants. In particular, it is necessary to identify from which TBox ($\mathcal{T}_\mathbf{l}$ or $\mathcal{T}_\mathbf{r}$) or concept (C or D) a concept in the content of a node is derived. After we have that information, we can extract an interpolant from a closed tableau.

The interpolant calculation rules are presented in Figure 2. Given a closed tableau $\mathbf{T}$ for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r}\rangle$, the interpolant calculation algorithm starts by calculating a concept $\mathsf{int}(g)$ for every sink node g in $\mathbf{T}$ with $g.\mathsf{status} = \texttt{unsat}$[2] using $\mathsf{C}_\bot$. While g_0 is not assigned a concept $\mathsf{int}(g_0)$, it repeatedly applies the following steps.

1. Pick a node g such that $\mathsf{int}(g)$ is undefined and $g.\mathsf{status} = \texttt{unsat}$.

[1] a node with no outgoing edges

[2] Note that a node g in $\mathbf{T}$ with $g.\mathsf{status} = \texttt{unsat}$ is a sink if and only if $\mathsf{R}_\bot$ is applied to g.

The $\mathsf{C}_\bot$ rule	**The $\mathsf{C}_\sqcup$ rule**
$\mathsf{int}(g) := \bot$, if $\lambda = \kappa = \mathbf{l}$; $\mathsf{int}(g) := C$, if $\lambda = \mathbf{l}$ and $\kappa = \mathbf{r}$; $\mathsf{int}(g) := \top$, if $\lambda = \kappa = \mathbf{r}$; $\mathsf{int}(g) := \dot{\neg} C$, if $\lambda = \mathbf{r}$ and $\kappa = \mathbf{l}$.	$\mathsf{int}(g) := \mathsf{int}(g') \sqcup \mathsf{int}(g'')$, if $\lambda = \mathbf{l}$; $\mathsf{int}(g) := \mathsf{int}(g') \sqcap \mathsf{int}(g'')$, if $\lambda = \mathbf{r}$.
The $\mathsf{C}_\sqcap$ rule	**The $\mathsf{C}_\exists$ rule**
$\mathsf{int}(g) := \mathsf{int}(g')$.	$\mathsf{int}(g) := \exists R_i.\mathsf{int}(g_i)$, if $\lambda_i = \mathbf{l}$; $\mathsf{int}(g) := \forall R_i.\mathsf{int}(g_i)$, if $\lambda_i = \mathbf{r}$ for some $i \in \{1, \ldots, n\}$.

Fig. 2. Interpolant calculation rules for $\mathcal{ALC}$.

2. If $g.\mathsf{type} = \texttt{and-node}$, and g has a direct successor g' (g_i for some $i \in \{1, \ldots, n\}$) with $\mathsf{int}(g')$ (resp. $\mathsf{int}(g_i)$) defined then apply $\mathsf{C}_\sqcap$ (resp. $\mathsf{C}_\exists$) .
3. If $g.\mathsf{type} = \texttt{or-node}$, and for all direct successors g', g'' of g we have that $\mathsf{int}(g')$ and $\mathsf{int}(g'')$ defined then apply $\mathsf{C}_\sqcup$.

This algorithm terminates in time polynomial in the size of $\mathbf{T}$ because all sink nodes with status `unsat` are reachable from g_0, and it is guaranteed to have nodes satisfying conditions 2 and 3 by the virtue of $g_0.\mathsf{status} = \texttt{unsat}$. The correctness of the algorithm is shown in two steps: first we show that our interpolant calculation rules are sound, i.e., they compute interpolants; second we show completeness, i.e., that we always find an interpolant. The proofs of these theorems are very similar to the corresponding ones we presented in [6]. Lemma 1 now follows straightforwardly from the termination and correctness of the interpolant calculation algorithm, and Theorem 5.

5 Beth Definability in Extensions of $\mathcal{ALC}$

In this section, we present a polynomial reduction from $\mathcal{SHI}$ KB satisfiability to $\mathcal{ALC}$ KB satisfiability. This reduction allows us to prove Beth definability and definitorial completeness properties for any extension of $\mathcal{ALC}$ with constructors from $\{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$. Ten Cate et al. [5] showed that these logics are definitorially complete but because of their model theoretic argument, they provide no information on the size of the resulting TBoxes. In this section, we establish a tight upper bound for explicit definitions in these logics.

We will proceed in three steps: first reduce $\mathcal{SHI}$-concept satisfiability w.r.t. a KB to the same problem in $\mathcal{ALCHI}$, then $\mathcal{ALCHI}$ to $\mathcal{ALCI}$, and finally $\mathcal{ALCI}$ to $\mathcal{ALC}$. All these reductions use the axiom schema instantiation technique [9] which is based on the idea of removing the constructor at hand by instantiating its corresponding (modal) axiom schema [10] for each concept in cl or a relevant concept closure, and adding these instances to the TBox to obtain an equi-satisfiable KB. The first and third of these reductions are given in [11] and [12], respectively. To the best of our knowledge, the second one has not beed used before.

Definition 9. *Let C_0 be a $\mathcal{SHI}$-concept and $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{SHI}$-KB. Then $\tau_{\mathcal{S}}(C_0, \mathcal{K})$ is defined as the $\mathcal{ALCHI}$ KB $(\mathcal{T} \cup \mathcal{T}', \mathcal{H}')$, where*

- $\mathcal{T}' = \{\forall R.C \sqsubseteq \forall S.\forall S.C \mid \forall R.C \in \mathsf{cl}(C_0, \mathcal{K}),\ S \sqsubseteq_{\mathcal{H}} R \text{ and } \mathsf{Trans}(R) \in \mathcal{H}\}$.
- $\mathcal{H}'$ *is obtained from* $\mathcal{H}$ *by removing all transitivity axioms.*

Theorem 6 ([11]). *A $\mathcal{SHI}$-concept C_0 is satisfiable w.r.t. a $\mathcal{SHI}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ if and only if C_0 is satisfiable w.r.t. the $\mathcal{ALCHI}$-KB $\tau_{\mathcal{S}}(C_0, \mathcal{K})$.*

Definition 10. *Let C_0 be an $\mathcal{ALCHI}$-concept and let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{ALCHI}$-KB. Then $\tau_{\mathcal{H}}(C_0, \mathcal{K})$ is defined as the $\mathcal{ALCI}$-KB $(\mathcal{T} \cup \mathcal{T}', \emptyset)$, where $\mathcal{T}' = \{\forall S.C \sqsubseteq \forall R.C \mid \forall S.C \in \mathsf{cl}(C_0, \mathcal{K}),\ R \sqsubseteq_{\mathcal{H}} S,$ and $R \neq S\}$.*

Theorem 7. *An $\mathcal{ALCHI}$-concept C_0 is satisfiable w.r.t. an $\mathcal{ALCHI}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ if and only if C_0 is satisfiable w.r.t. the $\mathcal{ALCI}$ KB $\tau_{\mathcal{H}}(C_0, \mathcal{K})$.*

Dealing with inverse roles is a bit more intricate because the signature of the original KB needs to be changed. Let the $\mathcal{ALC}$-concept $\zeta(C)$ be like the $\mathcal{ALCI}$-concept C except that every occurrence of each inverse role R^- in C is replaced with a new role name R^c; the renaming transformation $\zeta(\cdot)$ extends to TBoxes in the natural way. Moreover, let $\iota(\cdot)$ be the inverse of $\zeta(\cdot)$, i.e., $\iota(\zeta(C)) = C$.

Definition 11. *Let C_0 be an $\mathcal{ALCI}$-concept and let $\mathcal{T}$ be an $\mathcal{ALCI}$-TBox. Then $\tau_{\mathcal{I}}(C_0, \mathcal{T})$ is defined as the $\mathcal{ALC}$-TBox $\mathcal{T}_1 \cup \mathcal{T}_2$, where:*

1. $\mathcal{T}_1 = \zeta(\mathcal{T})$.
2. *$\mathcal{T}_2$ is the set of all concept inclusion axioms of the forms $C \sqsubseteq (\forall R.\exists R^c.C)$ and $C \sqsubseteq (\forall R^c.\exists R.C)$ such that C is in $\mathsf{cl}(\zeta(C_0), \mathcal{T}_1)$ and R is a role name in $\mathsf{rol}(C_0, \mathcal{T})$.*

Theorem 8 ([12]). *An $\mathcal{ALCI}$-concept C_0 is satisfiable w.r.t. an $\mathcal{ALCI}$-TBox $\mathcal{T}$ if and only if the $\mathcal{ALC}$-concept $\zeta(C_0)$ is satisfiable w.r.t. the $\mathcal{ALC}$-TBox $\tau_{\mathcal{I}}(C_0, \mathcal{T})$.*

The following theorem follows directly from Theorems 6, 7, and 8.

Theorem 9. *A $\mathcal{SHI}$-concept C_0 is satisfiable w.r.t. a $\mathcal{SHI}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ if and only if the $\mathcal{ALC}$-concept $\zeta(C_0)$ is satisfiable w.r.t. the $\mathcal{ALC}$-KB $\tau_{\mathcal{I}}(C_0, \tau_{\mathcal{H}}(C_0, \tau_{\mathcal{S}}(C_0, (\mathcal{T}, \mathcal{H}))))$.*

Except for the last one, all the reductions presented in this section preserve the signature of the given KB. Therefore, an explicit definition in the less expressive logic is also an explicit definition in the more expressive one. As for the last reduction, it is possible to reconstruct an explicit definition in $\mathcal{ALCI}$ from the one in $\mathcal{ALC}$ by replacing role names corresponding to inverse roles, as demonstrated by the following theorem.

Theorem 10. *Let $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ and let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{ALC}X$-KB. If an $\mathcal{ALC}X$-concept C is implicitly definable from $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$ under $\mathcal{K}$ then C is explicitly definable from Σ under $\mathcal{K}$, and the size of the explicit definition of C from Σ under $\mathcal{K}$ is at most exponential in $|\mathcal{T}| + |\mathcal{H}| + |C|$.*

Proof. Let $\mathcal{K}$ be a $\mathcal{SHI}$-KB and let C and D be $\mathcal{SHI}$-concepts. Furthermore, let $\mathcal{K}_\star = (\tau_{\mathcal{I}}(E, \tau_{\mathcal{H}}(E, \tau_{\mathcal{S}}(E, (\mathcal{T}, \mathcal{H})))))$, where $E = (C \sqcap \neg D) \sqcup (D \sqcap \neg C)$. We have the following chain of equivalences: (†) $\mathcal{K} \models C \equiv D \Leftrightarrow E$ is unsatisfiable w.r.t. $\mathcal{K}$ $\Leftrightarrow^{\text{[Theorem 9]}}$ $\zeta(E)$ is unsatisfiable w.r.t. $\mathcal{K}_\star \Leftrightarrow \mathcal{K}_\star \models \zeta(C) \equiv \zeta(D)$.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{ALCX}$-KB and let C be an $\mathcal{ALCX}$-concept such that C is implicitly definable from Σ under $\mathcal{K}$. Then $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$. Every $\mathcal{ALCX}$-concept and every $\mathcal{ALCX}$-KB is trivially a $\mathcal{SHI}$-concept and a $\mathcal{SHI}$-KB, respectively, and therefore, by (†), we have that $\mathcal{K}_\star \cup \widehat{\mathcal{K}_\star} \models \zeta(C) \equiv \widehat{\zeta(C)}$, where $\widehat{\cdot}$ is exactly like $\widetilde{\cdot}$ except that Σ is substituted by $\Sigma_\star = \Sigma \cup \{R^c \mid R^c \in \mathsf{rol}(\zeta(C), \mathcal{K}_\star) \text{ and } R \in \Sigma\}$. $\zeta(C)$ and $\mathcal{K}_\star$ are an $\mathcal{ALC}$-concept and $\mathcal{ALC}$-KB, respectively. By Theorem 1 we have that there is an explicit definition D of $\zeta(C)$ from $\Sigma_\star$ under $\mathcal{K}_\star$ the size of which is at most exponential in $|\tau_{\mathcal{I}}(C, \tau_{\mathcal{H}}(C, \tau_{\mathcal{S}}(C, (\mathcal{T}, \mathcal{H}))))| + |\zeta(C)|$. In other words, $\mathcal{K}_\star \models \zeta(C) \equiv D$ and the size of D is at most exponential in $|\mathcal{T}| + |\mathcal{H}| + |C|$. By (†), $\mathcal{K} \models C \equiv \iota(D)$, where $\iota(D)$ is an $\mathcal{ALCX}$-concept. But by Definition 4, $\iota(D)$ is an explicit definition of C from Σ under $\mathcal{K}$. □

6 Conclusion

In this paper, we revisited the problem of definitorial completeness, i.e., whether a given general TBox $\mathcal{T}$ in a DL $\mathscr{L}$ can be rewritten to an acyclic TBox $\mathcal{T}'$ in $\mathscr{L}$. $\mathcal{ALC}$ and every $\mathcal{ALCX}$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ were already known to be definitorially complete [5]. Our main contribution in this paper was to establish a tight exponential bound on the size of the resulting acyclic TBoxes.

Our results show that concept definitions can be written exponentially more succinctly in general TBoxes than in acyclic TBoxes for the logics we considered. Therefore, general concept inclusions may help increase the readability of a TBox/ontology. However, general concept inclusions introduce a lot of non-determinism to reasoning algorithms. If a general TBox is used for concept definitions and our DL is definitorially complete then this is not much of a restriction since an equivalent acyclic TBox can be obtained. This suggests the use of the general TBox for user friendliness and a precomputed acyclic counterpart for reasoning.

Definitorial completeness is a fragile property and not every DL enjoys it. For example, $\mathcal{ALCO}$ does not have this property and requires the @-operator from hybrid logics to become definitorially complete [5]. It is worthwhile to note that the algorithm for computing acyclic TBoxes is based on tableau, and thus it is suitable for optimised implementations. We leave as an open problem whether a tight upper bound for $\mathcal{SHIQ}$ can be obtained in a similar way.

References

1. Baader, F., Nutt, W.: Basic description logics. [13] 43–95
2. Donini, F.M.: Complexity of reasoning. [13] 96–136
3. Horrocks, I.: Implementation and optimization techniques. [13] 306–346
4. Beth, E.W.: On Padoa's methods in the theory of definitions. Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings **56** (1953) 330–339
5. Ten Cate, B., Conradie, W., Marx, M., Venema, Y.: Definitorially complete description logics. In Doherty, P., Mylopoulos, J., Welty, C.A., eds.: KR, AAAI Press (2006) 79–89
6. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over dboxes. In Boutilier, C., ed.: IJCAI. (2009) 923–925

7. Baader, F.: Description logic terminology. [13] 485–495
8. Goré, R., Nguyen, L.A.: Exptime tableaux for alc using sound global caching. In Calvanese, D. et al., eds.: Description Logics(2007)
9. Calvanese, D., Giacomo, G.D., Lenzerini, M., Nardi, D.: Reasoning in expressive description logics. In Handbook of Automated Reasoning, Elsevier and MIT Press (2001) 1581–1634
10. Blackburn, P., de Rijke, M., Venema, Y.: Modal logic. Cambridge University Press, New York, NY, USA (2001)
11. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
12. Calvanese, D., Giacomo, G.D., Rosati, R.: A note on encoding inverse roles and functional restrictions in alc knowledge bases. In Description Logics(1998)
13. Baader, F. et al., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)