

Reducing $SHIQ^-$ Description Logic to Disjunctive Datalog Programs

Ullrich Hustadt

Department of Computer Science
University of Liverpool
Liverpool, UK
U.Hustadt@csc.liv.ac.uk

Boris Motik

FZI Research Center for Information
Technologies at the University of Karlsruhe
Karlsruhe, Germany
motik@fzi.de

Ulrike Sattler

Department of Computer Science
University of Manchester
Manchester, UK
sattler@cs.man.ac.uk

Abstract

As applications of description logics proliferate, efficient reasoning with large ABoxes (sets of individuals with descriptions) becomes ever more important. Motivated by the prospects of reusing optimization techniques from deductive databases, in this paper, we present a novel approach to checking consistency of ABoxes, instance checking and query answering, w.r.t. ontologies formulated using a slight restriction of the description logic $SHIQ^-$. Our approach proceeds in three steps: (i) the ontology is translated into first-order clauses, (ii) TBox and RBox clauses are saturated using a resolution-based decision procedure, and (iii) the saturated set of clauses is translated into a disjunctive datalog program. Thus, query answering can be performed using the resulting program, while applying all existing optimization techniques, such as join-order optimizations or magic sets. Equally important, the resolution-based decision procedure we present is for unary coding of numbers worst-case optimal, i.e. it runs in EXPTIME.

Introduction

In recent years description logics have found their application in various fields of computer science, including, but not limiting to data integration, knowledge representation and ontology modeling for the Semantic Web. Many practical DL reasoners have been built and applied to practical problems. The experience shows that these systems perform well when computing the subsumption hierarchy: they use practicable, highly optimized tableau-based algorithms (Horrocks, Sattler, & Tobies 2000), which perform much better on practical problems than their ExpTime worst-case computational complexity suggests (Tobies 2001).

However, new applications, such as metadata management in the Semantic Web, require efficient query answering over large ABoxes, i.e., sets of ground facts. So far, attempts have been made to answer queries by a reduction to ABox consistency checking, which can then be solved by employing above mentioned tableau algorithms. From a theoretical point of view, this approach is quite elegant, but from a practical point of view, it has a significant drawback: as the number of ABox individuals increases, the performance may become quite poor. We believe that there are two main reasons

for this. Firstly, tableau-based algorithms treat all individuals separately, i.e., they do not group individuals together depending on common properties. Secondly, to answer a query, one usually does not need to consider all ABox information. Rather, only a small subset of the ABox usually suffices to compute the query answer. We find it difficult to modify the tableau search strategy to take into account the query in the search. These deficiencies have already been acknowledged by the research community, and certain optimization techniques for instance retrieval have already been developed (Haarslev & Möller 2002). However, the performance of query answering is still often not satisfactory.

Since techniques for reasoning in deductive databases are nowadays mature, we believe it makes sense to examine how to apply them to improve ABox reasoning in description logics. In this paper, we present a novel technique for reducing $SHIQ^-$ knowledge bases to disjunctive datalog programs, while preserving the semantics of the knowledge base. $SHIQ^-$ is a very expressive description logics which is at the core of OWL-DL, a variant of the Ontology Web Language (OWL), the current standard for ontology modeling in the Semantic Web. $SHIQ^-$ differs from $SHIQ$ (Horrocks, Sattler, & Tobies 2000) in the additional restriction that number restrictions are allowed only for roles not having subroles.

Our reduction to datalog does not mean that we suggest employing non-monotonic negation or minimal model reasoning. Rather, we consider the disjunctive datalog formalism useful since it allows for optimization techniques such as magic sets (Greco 2003) or join-order optimizations (Abiteboul, Hull, & Vianu 1995). More precisely, the reduction to disjunctive datalog addresses the above mentioned two points in the following way. Firstly, query answering in disjunctive datalog can be done by manipulating individuals in sets, and applying each inference rule to all individuals in a set at once, rather than to each individual separately. This enables the join order optimization, which, based on the database statistics, estimates the amount of work done for different join orders (Abiteboul, Hull, & Vianu 1995). The second point is addressed by means of the magic sets transformation (Beeri & Ramakrishnan 1987). Roughly speaking, the query is modified so that during its evaluation, a set of relevant facts is derived, and checking original query conditions is limited to this estimation. The magic sets transfor-

mation for disjunctive programs has been presented recently in (Greco 2003), along with the empirical evidence of its usefulness.

Our translation of \mathcal{SHIQ}^- knowledge bases into disjunctive datalog programs is based upon a basic superposition decision procedure for \mathcal{SHIQ}^- , which is interesting in its own right. Many resolution decision procedures for various classes of logics have already been devised, e.g. for the DL^* class (Nivelle, Schmidt, & Hustadt 2000) or for the (loosely) guarded fragment with equality (Ganzinger & de Nivelle 1999). However, even though many description logics are subsets of guarded fragments and number restrictions of \mathcal{SHIQ} can be translated into formulae with equality, we are not aware of any translation of \mathcal{SHIQ} into DL^* or the loosely guarded fragment with equality.

Our decision procedure is based on basic superposition, a sophisticated clausal calculus for logics with equality (Bachmair *et al.* 1995). To show termination, we combine it with eager elimination of redundant clauses by subsumption. Interestingly, we employ subsumption to restrict the *depth* of clauses considered, whereas similar similar procedures typically employ subsumption to restrict the clause *length*.

Furthermore, our decision procedure runs in worst-case exponential time, and is thus optimal under the assumption of unary coding of numbers in the input. Such an assumption is quite common in description logics, even though \mathcal{SHIQ} is EXPTIME -complete regardless of the coding of numbers (Tobies 2001). In practice, this means that one should avoid large numbers in number restrictions. Based on the vast experience in building efficient theorem provers, we expect this procedure to be practicable.

Due to a lack of space, for the proofs of some results in this paper, we refer the interested reader to the accompanying technical report (Hustadt, Motik, & Sattler 2003).

Preliminaries

Description Logic \mathcal{SHIQ} . The syntax of \mathcal{SHIQ} is given by the following definition.

Definition 1. Let N_R be the set of role names. The set of \mathcal{SHIQ} roles is the set $N_R \cup \{R^- \mid R \in N_R\}$. For $R \in N_R$, let $\text{Inv}(R)$ denote R^- and let $\text{Inv}(R^-)$ denote R . An RBox \mathcal{R} over N_R is a finite set of transitivity axioms $\text{Trans}(R)$ and role inclusion axioms $R \sqsubseteq S$, where R and S are roles, such that, if $R \sqsubseteq S \in \mathcal{R}$, then $\text{Inv}(R) \sqsubseteq \text{Inv}(S) \in \mathcal{R}$ as well. Let \sqsubseteq^* denote the reflexive-transitive closure of \sqsubseteq . A role R is transitive if $\text{Trans}(S) \in \mathcal{R}$ or $\text{Trans}(\text{Inv}(S)) \in \mathcal{R}$ for some S with $S \sqsubseteq^* R$ and $R \sqsubseteq^* S$; R is simple if there is no role S such that $S \sqsubseteq^* R$ and S is transitive; R is complex if it is not simple.

Let N_C be a set of atomic concept names. The set of \mathcal{SHIQ} concepts over N_C and N_R is defined inductively as the smallest set for which the following holds: \top and \perp are \mathcal{SHIQ} concepts, each atomic concept name $A \in N_C$ is a \mathcal{SHIQ} concept, if C and D are \mathcal{SHIQ} concepts and R is a role, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$ are also \mathcal{SHIQ} concepts, and, if C is a \mathcal{SHIQ} concept, R a simple role and n an integer, then $\leq n R.C$ and $\geq n R.C$ are \mathcal{SHIQ} concepts.

$\text{TBox } \mathcal{T}$ over N_C and \mathcal{R} is a finite set of concept inclusion axioms $C \sqsubseteq D$ or concept equivalence axioms $C \equiv D$, where C and D are \mathcal{SHIQ} concepts.

Let N_I be a set of individual names. An $\text{ABox } \mathcal{A}$ is a set of concept and role membership axioms $C(a)$ and $R(a, b)$, and (in)equality axioms $a \approx b$ and $a \not\approx b$, where C is a \mathcal{SHIQ} concept, R a role, and a and b are individuals.

A \mathcal{SHIQ} knowledge base KB is a triple of the form $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$, where $KB_{\mathcal{R}}$ is an RBox , $KB_{\mathcal{T}}$ is a TBox , and $KB_{\mathcal{A}}$ is an ABox .

Please note that we do not assume the unique names assumption, but allow the user to axiomatize it explicitly using inequalities, cf. (Baader *et al.* 2003, p. 60).

Definition 2. The semantics of a \mathcal{SHIQ} knowledge base KB is given by the mapping π which transforms KB axioms into a set of first-order formulae, as presented in Table 1. We call KB satisfiable if $\pi(KB)$ is satisfiable.

Other interesting inference problems can be reduced to satisfiability as follows, where α denotes a new individual not occurring in KB :

Concept satisfiability. A concept C is satisfiable w.r.t. KB if there exists a model of KB in which the interpretation of C is not empty. This is the case iff $KB \cup C(\alpha)$ is satisfiable.

Subsumption. A concept C is subsumed by a concept D w.r.t. KB if $\pi(KB) \models \pi(C \sqsubseteq D)$. This is the case iff $KB \cup (C \sqcap \neg D)(\alpha)$ is unsatisfiable.

Instance checking. An individual i is an instance of a concept C w.r.t. KB if $\pi(KB) \models \pi(C(i))$. This is the case iff $KB \cup \neg C(i)$ is unsatisfiable.

We now define a slight restriction of \mathcal{SHIQ} description logic to which the approach in this paper is applicable.

Definition 3. For a knowledge base KB , a role R is called very simple if no role S different from R exists, such that $S \sqsubseteq^* R \in KB_{\mathcal{R}}$. The description logic \mathcal{SHIQ}^- is the fragment of \mathcal{SHIQ} obtained by restricting number restrictions $\leq n R.C$ and $\geq n R.C$ to very simple roles R .

We also consider the \mathcal{ALCHIQ}^- fragment of \mathcal{SHIQ}^- , which does not allow transitivity axioms.

Basic Superposition Calculus. The basic superposition calculus has been developed to optimize theorem proving with equality (Bachmair *et al.* 1995). A similar calculus was developed by Nieuwenhuis and Rubio (Nieuwenhuis & Rubio 1995).

We assume a standard notion of first-order clauses with equality: all existential quantifiers have been eliminated using Skolemization; all remaining variables are universally quantified; we only consider the equality predicate (all non-equational literals A are encoded as $A \approx \top$ in a multi-sorted setting); and we treat \approx as having built-in symmetry. Moreover, we assume the reader to be familiar with standard first-order resolution (Bachmair & Ganzinger 2001).

Basic superposition is an optimized version of superposition (a calculus for equational theories (Bachmair &

Mapping Concepts to FOL	
$\pi_y(\top, X) = \top$	$\pi_y(\perp, X) = \perp$
$\pi_y(A, X) = A(X)$	$\pi_y(\neg C, X) = \neg \pi_y(C, X)$
$\pi_y(C \sqcap D, X) = \pi_y(C, X) \wedge \pi_y(D, X)$	$\pi_y(C \sqcup D, X) = \pi_y(C, X) \vee \pi_y(D, X)$
$\pi_y(\forall R.C, X) = \forall y : R(X, y) \rightarrow \pi_x(C, y)$	$\pi_y(\exists R.C, X) = \exists y : R(X, y) \wedge \pi_x(C, y)$
$\pi_y(\leq n R.C, X) = \forall y_1, \dots, y_{n+1} : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \rightarrow \bigvee y_i \approx y_j$	
$\pi_y(\geq n R.C, X) = \exists y_1, \dots, y_n : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \wedge \bigwedge y_i \not\approx y_j$	
Mapping Axioms to FOL	
$\pi(C \sqsubseteq D) = \forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$	
$\pi(C \equiv D) = \forall x : \pi_y(C, x) \leftrightarrow \pi_y(D, x)$	
$\pi(R \sqsubseteq S) = \forall x, y : R(x, y) \rightarrow S(x, y)$	
$\pi(\text{Trans}(R)) = \forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$	
Mapping KB to FOL	
$\pi(R) = \forall x, y : R(x, y) \leftrightarrow R^-(y, x)$	
$\pi(KB_{\mathcal{R}}) = \bigwedge_{\alpha \in KB_{\mathcal{R}}} \pi(\alpha) \wedge \bigwedge_{R \in N_{\mathcal{R}}} \pi(R)$	
$\pi(KB_{\mathcal{T}}) = \bigwedge_{\alpha \in KB_{\mathcal{T}}} \pi(\alpha)$	
$\pi(KB_{\mathcal{A}}) = \bigwedge_{C(a) \in KB_{\mathcal{A}}} \pi_y(C, a) \wedge \bigwedge_{R(a,b) \in KB_{\mathcal{A}}} R(a, b) \wedge \bigwedge_{a \approx b \in KB_{\mathcal{A}}} a \approx b \wedge \bigwedge_{a \not\approx b \in KB_{\mathcal{A}}} a \not\approx b$	
$\pi(KB) = \pi(KB_{\mathcal{R}}) \wedge \pi(KB_{\mathcal{T}}) \wedge \pi(KB_{\mathcal{A}})$	
where X is a meta variable and is substituted by the actual variable and π_x is defined as π_y by substituting x and x_i for all y and y_i , respectively.	

Table 1: Translation of \mathcal{SHIQ} into FOL

Ganzinger 1994)) which prohibits superposition into terms introduced by previous unification steps, thus reducing the number of clauses generated. Its inferences rules are formalized by distinguishing two parts of a clause: (i) the *skeleton* clause C and (ii) the *substitution* σ representing the cumulative effects of previous unifications. Such a representation of $C\sigma$ is called a *closure*, and is written as $C \cdot \sigma$. A closure can conveniently be represented by *marking* the terms in $C\sigma$ occurring at variable positions of C . Any position at or beneath a marked position is called a *substitution position*. E.g., the clause $P(f(y)) \vee g(b) \approx b$ is logically equivalent to the closure $(P(x) \vee z \approx b) \cdot \{x \mapsto f(y), z \mapsto g(b)\}$, which can conveniently be represented as $P([f(y)]) \vee [g(b)] \approx b$.

The calculus requires two parameters. The first is an *admissible* ordering on terms \succ , i.e., a *reduction ordering* total on ground terms. Such an ordering is then extended to literals (Bachmair *et al.* 1995). The second parameter of the calculus is a *selection function* which selects an arbitrary set of negative literals in a closure.

Given these parameters, the basic superposition calculus, \mathcal{BS} for short, consists of the inference rules given in Table 2. A *derivation* from a closure set N_0 is a sequence of closure sets N_0, N_1, \dots, N_i , where $N_i = N_{i-1} \cup \{C\}$, and C is derived by applying an inference rule from Table 2 with premises from N_{i-1} . Roughly speaking, the set of closures N_i is *saturated up to redundancy* if all inferences from premises in N_i are redundant in N_i . If this is the case, then N_i contains the empty closure if and only if N_0 is unsatisfiable, so \mathcal{BS} is a sound and complete refutation procedure (Bachmair *et al.* 1995). Compatible redundancy elimination rules have been presented in (Bachmair *et al.* 1995; Hustadt, Motik, & Sattler 2003)

Algorithm Overview

Our algorithm for reducing a \mathcal{SHIQ}^- knowledge base to a disjunctive datalog program $\text{DD}(KB)$ proceeds by the fol-

lowing five steps, each of which preserves satisfiability and entailment of ground facts of the form $C(a)$ and $R(a, b)$, where R is a simple role:

1. KB is translated first into \mathcal{ALCHIQ}^- knowledge base $\Omega(KB)$ by eliminating transitivity axioms. The size of $\Omega(KB)$ is polynomial in $|KB|$.
2. Then, $\Omega(KB)$ is translated into a first-order formula, using the operator π from Table 1, which is then transformed into a set of clauses $\Xi(KB)$. We use structural transformation for the clausal transformation (Nonnengart & Weidenbach 2001; Nivelle, Schmidt, & Hustadt 2000), which is known to be polynomial.
3. \mathcal{BS} is applied to $\Gamma_{\mathcal{TR}} = \Xi(KB_{\mathcal{T}} \cup KB_{\mathcal{R}})$, that is, to TBox and RBox clauses of KB . We show that this yields clauses of a rather restricted form only, so-called \mathcal{ALCHIQ}^- -clauses, presented in Table 3. The saturated set $\text{Sat}(\Gamma_{\mathcal{TR}})$ is of size exponential in $|KB|$.
4. In case $\text{Sat}(\Gamma_{\mathcal{TR}})$ does not contain the empty clause, $\text{Sat}(\Gamma_{\mathcal{TR}}) \cup \Xi(KB_{\mathcal{A}})$ is translated into a function-free version $\text{FF}(KB)$, in which each ground functional term $f(a)$ is simulated with a new constant a_f . We show that $\text{FF}(KB)$ is of size exponential in $|KB|$.
5. Since it is function-free, $\text{FF}(KB)$ can be transformed into a positive disjunctive datalog program with equality $\text{DD}(KB)$. The program $\text{DD}(KB)$ is of size exponential in $|KB|$.

The program $\text{DD}(KB)$ can be used for query answering, i.e., $KB \models C(a)$ iff $\text{DD}(KB) \models \pi(C(a))$; and for a simple role R , $KB \models R(a, b)$ iff $\text{DD}(KB) \models \pi(R(a, b))$. For TBox reasoning tasks, such as computing the subsumption hierarchy, $\text{Sat}(\Gamma_{\mathcal{TR}})$ suffices, i.e., $KB \models C \sqsubseteq D$ iff $\text{Sat}(\Gamma_{\mathcal{TR}}) \models \pi(C \sqsubseteq D)$.

Positive superposition: $\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \approx v) \cdot \rho}{(C \vee D \vee w[t]_p \approx v) \cdot \theta}$	(i) $\sigma = \text{MGU}(s\rho, w\rho _p)$, (ii) $\theta = \rho\sigma$, (iii) $t\theta \not\approx s\theta$ and $v\theta \not\approx w\theta$, (iv) $(s \approx t) \cdot \theta$ is SES, (v) $(w \approx v) \cdot \theta$ is SES, (vi) $s\theta \approx t\theta \not\approx w\theta \approx v\theta$, (vii) $w _p$ is not a variable.
Negative superposition: $\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \not\approx v) \cdot \rho}{(C \vee D \vee w[t]_p \not\approx v) \cdot \theta}$	(i) $\sigma = \text{MGU}(s\rho, w\rho _p)$, (ii) $\theta = \rho\sigma$, (iii) $t\theta \not\approx s\theta$ and $v\theta \not\approx w\theta$, (iv) $(s \approx t) \cdot \theta$ is SES, (v) $(w \not\approx v) \cdot \theta$ is ER, (vi) $w _p$ is not a variable.
Reflexivity resolution: $\frac{(C \vee s \not\approx t) \cdot \rho}{C \cdot \theta}$	(i) $\sigma = \text{MGU}(s\rho, t\rho)$, (ii) $\theta = \rho\sigma$, (iii) $(s \not\approx t) \cdot \theta$ is ER.
Equality factoring: $\frac{(C \vee s \approx t \vee s' \approx t') \cdot \rho}{(C \vee t \not\approx t' \vee s' \approx t') \cdot \theta}$	(i) $\sigma = \text{MGU}(s\rho, s'\rho)$, (ii) $\theta = \rho\sigma$, (iii) $t\theta \not\approx s\theta$ and $t'\theta \not\approx s'\theta$, (iv) $(s \approx t) \cdot \theta$ is ES.
Ordered resolution: $\frac{(C \vee A) \cdot \rho \quad (D \vee \neg B) \cdot \rho}{(C \vee D) \cdot \theta}$	(i) $\sigma = \text{MGU}(A\rho, B\rho)$, (ii) $\theta = \rho\sigma$, (iii) $A \cdot \theta$ is SES, (iv) $\neg B \cdot \theta$ is ER.
Notes:	
(i) $L \cdot \sigma$ is maximal in $C \cdot \sigma$ if there is no $L' \in C \setminus \{L\}$ such that $L'\sigma \succ L\sigma$.	
(ii) $L \cdot \sigma$ is strictly maximal in $C \cdot \sigma$ if there is no $L' \in C \setminus \{L\}$ such that $L'\sigma \succeq L\sigma$.	
(iii) A literal $L \cdot \theta$ is (strictly) eligible for superposition ((S)ES) in a closure $(C \vee L) \cdot \theta$ if nothing is selected in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is (strictly) maximal in $C \cdot \theta$.	
(iv) A literal $L \cdot \theta$ is eligible for resolution (ER) in a closure $(C \vee L) \cdot \theta$ if it is selected in $(C \vee L) \cdot \theta$ or nothing is selected in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is maximal in $C \cdot \theta$.	

Table 2: Inference Rules of Basic Superposition

Eliminating Transitivity Axioms

In this section we show how to eliminate transitivity axioms from a \mathcal{SHIQ} knowledge base by transforming it into an equi-satisfiable \mathcal{ALCHIQ} knowledge base. A similar transformation may be found in (Tobies 2001), where an algorithm for transforming \mathcal{SHIQ} concepts to concepts in a related \mathcal{ALCQ} logic was presented. Another similar transformation has been presented in (Schmidt & Hustadt 2003), where it is demonstrated, among others, how to encode multi-modal logic with transitive modalities $\mathbf{K4}_m$ into plain multi-modal logic \mathbf{K}_m . In the following, we use $\text{NNF}(C)$ to denote the negation-normal form of C (Horrocks, Sattler, & Tobies 2000).

Definition 4. For some \mathcal{SHIQ} knowledge base KB , let $\text{clos}(KB)$ denote the concept closure of KB , defined as the smallest set of concepts satisfying the following conditions:

- $C \sqsubseteq D \in KB_{\mathcal{T}}$ implies $\text{NNF}(\neg C \sqcup D) \in \text{clos}(KB)$.
- $C(a) \in KB_{\mathcal{A}}$ implies $\text{NNF}(C) \in \text{clos}(KB)$.

- $C \in \text{clos}(KB)$ and D being a sub-concept of C implies $D \in \text{clos}(KB)$.
- $\forall R.C \in \text{clos}(KB)$, $S \sqsubseteq^* R$, and $\text{Trans}(S) \in KB_{\mathcal{R}}$ implies $\forall S.C \in \text{clos}(KB)$.

Please note that all concepts in $\text{clos}(KB)$ are in NNF. Now we define the operator Ω which translates a \mathcal{SHIQ} knowledge base KB into an \mathcal{ALCHIQ} knowledge base $\Omega(KB)$.

Definition 5. For a \mathcal{SHIQ} knowledge base KB , let $\Omega(KB)$ denote the following \mathcal{ALCHIQ} knowledge base:

- $\Omega(KB)_{\mathcal{R}}$ is obtained from $KB_{\mathcal{R}}$ by removing all axioms $\text{Trans}(R)$,
- $\Omega(KB)_{\mathcal{T}} = KB_{\mathcal{T}} \cup \{\forall R.C \sqsubseteq \forall S.(\forall S.C) \mid \forall R.C \in \text{clos}(KB) \wedge S \sqsubseteq^* R \wedge \text{Trans}(S) \in KB_{\mathcal{R}}\}$,
- $\Omega(KB)_{\mathcal{A}} = KB_{\mathcal{A}}$.

Observe that, for any concept C , the number of subconcepts in $\text{clos}(KB)$ is bounded by the number of subexpressions in C . Furthermore, for each concept from $\text{clos}(KB)$, we may generate at most $|N_{\mathcal{R}}|$ axioms in $\Omega(KB)_{\mathcal{T}}$. Hence, the encoding is polynomial in $|KB|$. Furthermore, in a way similar to the one found in (Tobies 2001), we show in (Hustadt, Motik, & Sattler 2003) that this encoding does not affect satisfiability.

Theorem 1. KB is satisfiable iff $\Omega(KB)$ is satisfiable.

Notice that $\Omega(KB \cup \{(\neg)C(a)\}) = \Omega(KB) \cup \{(\neg)C(a)\}$, so $KB \models (\neg)C(a)$ iff $\Omega(KB) \models (\neg)C(a)$. However, the models of KB and $\Omega(KB)$ may differ in the interpretation of complex roles, so $\Omega(KB)$ can be used only to prove entailment of ground facts $(\neg)R(a, b)$ for a simple role R .

Deciding \mathcal{ALCHIQ}^- by Basic Superposition

In this section, we show how to decide satisfiability of an \mathcal{ALCHIQ}^- knowledge base KB . We assume that, for all facts $C(a) \in KB_{\mathcal{A}}$, C is an atomic concept. This is without loss of generality since each $C(a) \in KB_{\mathcal{A}}$ where C is not atomic can be replaced with a pair of axioms $A_C(a)$, $A_C \sqsubseteq C$ for A_C a new concept name while preserving the semantics. This transformation is obviously polynomial.

Preprocessing. The first step in deciding satisfiability of KB is to transform it into clausal form. In order to avoid the exponential blow-up by direct clausification of $\pi(KB)$, we apply the well-known *structural transformation* (Nonnengart & Weidenbach 2001). Let φ be some formula in negation-normal form, and Λ a subset of positions of subformulae of φ . By $\text{Def}_{\Lambda}(\varphi)$ we denote the *definitional normal form* of φ with respect to Λ , computed as explained in (Nonnengart & Weidenbach 2001). Furthermore, let $\text{Cls}(\varphi)$ denote the set of closures obtained by the usual clausification by structural skolemization (Nonnengart & Weidenbach 2001). It is well-known that, if φ does not contain nested equivalences, then $\text{Cls}(\text{Def}_{\Lambda}(\varphi))$ can be computed in polynomial time. Furthermore, φ is satisfiable if and only if $\text{Cls}(\text{Def}_{\Lambda}(\varphi))$ is.

Let $\Xi(KB) = \text{Cls}(\text{Def}_{\Lambda}(\pi(KB)))$, where Λ is the set of non-literal subformulae positions of $\pi(KB)$. It is easy to

see that all closures in $\Xi(KB)$ share some common syntactic properties. Table 3 lists the types of so-called $\mathcal{ALCCHI}Q^-$ -closures of $\Xi(KB)$. We use $\mathbf{P}(x)$ to denote a possibly empty disjunction $(\neg)P_1(x) \vee \dots \vee (\neg)P_n(x)$; $\mathbf{P}(f(x))$ for a possibly empty disjunction $\mathbf{P}_1(f_1(x)) \vee \dots \vee \mathbf{P}_n(f_n(x))$; and $\langle t \rangle$ to express that t may, but need not be marked.

Lemma 1. *Each closure from $\Xi(KB)$ is of exactly one type from Table 3. Also, for each function symbol f occurring in $\Xi(KB)$, there is exactly one closure of type 3 containing $f(x)$ unmarked; this closure is called the R^f -generator, the disjunction $\mathbf{P}^f(x)$ is called the f -support, and R is called the designated role for f and is denoted as $\text{role}(f)$.*

Proof. The first claim follows trivially from the definition of $\Xi(KB)$. Furthermore, each closure of type 3 is generated by skolemizing an existentially quantified subformula by introducing a fresh function symbol, so each function symbol is associated with exactly one closure of type 3. \square

Parameters for Basic Superposition. We now specify our parameters for basic superposition.

Definition 6. *We use \mathcal{BS}_{DL} for the calculus \mathcal{BS} parameterized as follows: (i) the term ordering \succ is a lexicographic path ordering (LPO) (Bachmair & Ganzinger 2001) induced by a total precedence $>_P$ on function, constant and predicate symbols, such that, for any function symbol f , constant symbol c , and predicate symbol p , we have $f >_P c >_P p >_P \top$; and (ii) the selection function selects, in each closure $C \cdot \sigma$, every negative binary literal.*

In \mathcal{BS}_{DL} , we need to compare terms and literals only in closures of types 3–6 and 9 from Table 3. Since LPOs are total on ground terms, and terms in closures of type 3–6 and 9 have at most one variable, any LPO is total on non-ground terms from these closures. In this case, we compare literals by associating, with each literal L , the complexity measure $c_L = (\max(L), p_L, \min(L))$, where $\max(L)$ ($\min(L)$) is the maximum (minimum) of the two terms in L , and p_L is 1 if L is negative, and 0 otherwise. Then $L_1 \succ L_2$ if and only if $c_{L_1} \succ c_{L_2}$, where c_L are compared lexicographically, by using the LPO \succ on terms to compare the first and the third positions, and taking 1 \succ 0 for the second position. It is easy to see that this definition of the literal ordering is compatible with the one from (Bachmair *et al.* 1995).

Furthermore, observe that, if $s \succ t$, then, for any substitution σ , obviously $s\sigma \succ t\sigma$. Hence, since LPOs are total on terms in closures of type 3–6 and 9, any two terms can always be compared, so the ordering and selection constraints can be checked *a priori*, that is, before computing the unifier, which is much easier to implement in practice.

Closure of $\mathcal{ALCCHI}Q^-$ -closures under Inferences. The following lemma lies at the heart of our decision procedure.

Lemma 2. *Let $\Xi(KB) = N_0, \dots, N_i \cup \{C\}$ be a \mathcal{BS}_{DL} -derivation, where C is the conclusion derived from premises in N_i . Then C is either an $\mathcal{ALCCHI}Q^-$ -closure or is redundant.*

Proof. (Sketch) The proof is by induction on the derivation length where, in the induction step, we consider all possible applications of all inference rules. Most importantly, we show that we never obtain functional terms of depth greater than 2. This is due to the properties of LPOs, the choice of our precedence $>_P$, and the fact that, in each closure, all terms of depth two are of the form $f_i(g(x))$ and, if such a term occurs in a closure, all terms of depth one occurring in this closure are of the form $g(x)$. These three facts ensure that (i) the maximal literal L of a closure is also of maximal depth, which ensures that, after unification, $L\sigma$ is still of maximal depth; and (ii) since we only have unary functions, if two terms $f_1(\dots f_k(x) \dots)$ and $g_1(\dots g_\ell(y) \dots)$ are unifiable, then $f_1 \dots f_k$ is a prefix of $g_1 \dots g_\ell$ or vice versa, and thus unification of two terms s, t yields terms of depth bounded by the maximum depth of s and t .

Resolution is only applicable to a closure of type 3, 4, or 9 with a closure of type 1, 2, 7, or 8, or between closures of type 5, 6, or 9. From the observations above, resolution will never result in a closure with a term of depth greater than two. Similarly, it is easy to show that reflexivity resolution and equality factoring will always produce an $\mathcal{ALCCHI}Q^-$ -closure. The only inferences which are more involved are positive and negative superposition.

Since number restrictions in $\mathcal{ALCCHI}Q^-$ are restricted to very simple roles, in all literals of the form $[f_1(x)] \approx [f_2(x)]$ or $[f_1(g(x))] \approx [f_2(g(x))]$, f_1 and f_2 have the same designated role. Similarly, all literals of the form $[f(g(x))] \approx x$ are generated by resolving an R^f -generator with a closure of type 4, obtained by resolving a $\text{Inv}(\text{role}(f))^g$ -generator with a closure of type 1. Since all functional terms are marked in all literals considered here, *basic* positive or negative superposition cannot be applied to them. Hence, conditions (5.ii), (6.v), (6.vi), (7.viii), (8.vii), (8.viii), (9.iv) and (9.v) are always preserved.

It is easy to see that, if some closure contains $f(\iota)$, it also contains $\mathbf{P}^f(\iota)$. This invariant is the consequence of the fact that the literal containing $f(\iota)$ is greater than any literal from $\mathbf{P}^f(\iota)$, so no inference with any $\mathbf{P}^f(\iota)$ can take place. Hence, conditions (5.i), (6.iii), (6.iv), (7.vii), (8.v), (8.vi) and (9.iii) are always preserved.

In all terms of the form $f(g(x))$, the subterm $g(x)$ occurs always marked, so superposition is allowed only at the outer-most position. Hence, all superposition inferences into a closure of type 5, 6, or 9, produce a closure of type 5, 6, or 9. The only other possible superpositions are into a closure $(D \vee w \approx v) \cdot \rho$ of type 3, say with a free variable x' . By the ordering constraints, superposition is possible only into $R(x', f(x'))$. If superposition is from a literal $[f(\iota)] \approx [g(\iota)]$, the unifier σ is $\{x' \mapsto \iota\}$, and the conclusion is $\mathbf{P}^f([\iota]) \vee R([\iota], [g(\iota)]) \vee C \cdot \rho$ where $C \cdot \rho$ contains $\mathbf{P}^g([\iota])$. However, by conditions (5.ii), (6.v) and (6.vi), the R^g -generator of the form $\mathbf{P}^g(y) \vee R(y, g(y))$ exists, and it subsumes the inference conclusion via substitution $\{y \mapsto \iota\}$, so this inference is redundant. In a similar way, one can show that, for any superposition into a closure of type 3, there is always a closure which subsumes the conclusion, so any such superposition is redundant. \square

1	$\neg R(x, y) \vee \text{Inv}(R)(y, x)$
2	$\neg R(x, y) \vee S(x, y)$
3	$\mathbf{P}^f(x) \vee R(x, \langle f(x) \rangle)$
4	$\mathbf{P}^f(x) \vee R(\langle f(x) \rangle, x)$
5	$\mathbf{P}_1(x) \vee \mathbf{P}_2(\langle f(x) \rangle) \vee \bigvee [f_i(x)] \approx [f_j(x)] \vee \bigvee \langle f_i(x) \rangle \not\approx \langle f_j(x) \rangle$ (i): for each $f_i(x)$ the closure contains $\mathbf{P}^{f_i}(x)$, (ii): for each $[f_i(x)] \approx [f_j(x)]$ we have $\text{role}(f_i) = \text{role}(f_j)$.
6	$\mathbf{P}_1(x) \vee \mathbf{P}_2(\langle g(x) \rangle) \vee \mathbf{P}_3(\langle t \rangle) \vee \bigvee [t_i] \approx [t_j] \vee \bigvee \langle t_i \rangle \not\approx \langle t_j \rangle$ (i): there is at least one term of the form $f_i(\langle g(x) \rangle)$, (ii): terms t, t_i and t_j are of the form x or $f_i(\langle g(x) \rangle)$, (iii): for each $f_i(\langle g(x) \rangle)$, the closure contains $\mathbf{P}^{f_i}(\langle g(x) \rangle)$, (iv): the closure contains $\mathbf{P}^g(x)$, (v): for each $[f_i(\langle g(x) \rangle)] \approx [f_j(\langle g(x) \rangle)]$, we have $\text{role}(f_i) = \text{role}(f_j)$, (vi): for each $[f_i(\langle g(x) \rangle)] \approx x$, there is a closure $P^g(x) \vee \text{role}(f_i)(\langle g(x) \rangle, x)$.
7	$\bigvee \neg R(\langle u \rangle, y_i) \vee \mathbf{P}_1(\mathbf{y}) \vee \mathbf{P}_2(x) \vee \mathbf{P}_3(\langle f(x) \rangle) \vee \bigvee [t_i] \approx [t_j] \vee G$ (i): there is at least one literal $\neg R(\langle u \rangle, y_i)$, (ii): terms t_i and t_j are of the form y_i, x or a constant c , or a term $f(\langle u \rangle)$, (iii): u is the variable x or u is a constant and x does not appear in the closure, (iv): each y_i occurs as the second argument of exactly one $\neg R(\langle u \rangle, y_i)$, (v): for each pair of variables y_i and y_j , there is a literal $y_i \approx y_j$, (vi): G is a closure of type 9, (vii): for each $f_i(u)$, the closure contains $\mathbf{P}^{f_i}(\langle u \rangle)$, (viii): for each $[f_i(u)] \approx [f_j(u)]$, we have $\text{role}(f_i) = \text{role}(f_j)$.
8	$\bigvee \neg R(\langle g(x) \rangle, y_i) \vee \mathbf{P}_1(\mathbf{y}) \vee \mathbf{P}_2(x) \vee \mathbf{P}_3(\langle g(x) \rangle) \vee \mathbf{P}_4(\langle f(g(x)) \rangle) \vee \bigvee [t_i] \approx [t_j]$ (i): there is at least one literal $\neg R(\langle g(x) \rangle, y_i)$, (ii): terms t_i and t_j are of the form y_i, x or $f_i(\langle g(x) \rangle)$, (iii): each y_i occurs as the second argument of exactly one $\neg R(\langle g(x) \rangle, y_i)$, (iv): for variable y_i , there is a literal $y_i \approx x$, (v): for each $f_i(\langle g(x) \rangle)$ the closure contains $\mathbf{P}^{f_i}(\langle g(x) \rangle)$, (vi): the closure contains $\mathbf{P}^g(x)$, (vii): for each $[f_i(\langle g(x) \rangle)] \approx [f_j(\langle g(x) \rangle)]$, we have $\text{role}(f_i) = \text{role}(f_j)$, (viii): for each $[f_i(\langle g(x) \rangle)] \approx x$, there is a closure $P^g(x) \vee \text{role}(f_i)(\langle g(x) \rangle, x)$.
9	$\mathbf{R}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle) \vee \mathbf{P}_1(\langle \mathbf{a} \rangle) \vee \mathbf{P}_2(\langle \mathbf{f}(\mathbf{a}) \rangle) \vee \bigvee \langle t_i \rangle \approx \langle t_j \rangle \vee \bigvee \langle t_i \rangle \not\approx \langle t_j \rangle$ (i): terms t_i and t_j are of the form a or $f_i(\langle a \rangle)$, (ii): equality literals may contain only constant terms non-marked, (iii): for each $f_i(\langle a \rangle)$ the closure contains $\mathbf{P}^{f_i}(\langle a \rangle)$, (iv): for each $[f_i(\langle a \rangle)] \approx [f_j(\langle a \rangle)]$, we have $\text{role}(f_i) = \text{role}(f_j)$, (v): for each $[f_i(\langle a \rangle)] \approx \langle b \rangle$ there is an $R(a, b)$ -witness $R(\langle a \rangle, \langle b \rangle) \vee D \cdot \sigma$, where $D \cdot \sigma$ does not contain functional terms, it is contained in this closure, and $R = \text{role}(f_i)$.

Table 3: Types of \mathcal{ALCHIQ}^- -closures

A slight optimization is possible. Namely, any closure of type 7 with n binary literals can be resolved with n premises in $n!$ ways. However, closures of type 7 in $\Xi(KB)$ are symmetric with respect to variables y_i , so all of the $n!$ resolutions will result in the same closure. Obviously, this can be optimized by ordering the premises and performing just one resolution. We formalize this idea by attaching a constraint $T = y_1 \succ \dots \succ y_n$ to closures of type 7 in $\Xi(KB)$ and resolving binary literals in closures of type 7 and 8 from left to right. Each time a closure of type 7 participates in a resolution with unifier σ , we compute $T\sigma$. If ordering constraints are not satisfied, the conclusion is deleted; otherwise, the constraint $T\sigma$ is attached to it.

Lemma 3. *The constraint inheritance explained above does not affect soundness or completeness of \mathcal{BS}_{DL} .*

Termination and Complexity Analysis. We now show that \mathcal{BS}_{DL} terminates on \mathcal{ALCHIQ}^- . Let $|KB|$ be the

size of the knowledge base, measured as the number of symbols needed to encode KB on the input tape of a Turing machine, by using a single symbol for each atomic concept, role and individual. For each syntactic construct of KB , its size can be computed recursively by adding up the sizes of all parts. Furthermore, we assume *unary* coding of numbers, so $|\geq n R.C| = |\leq n R.C| = n + 2 + |C|$.

Lemma 4. *Let N_i be any closure set obtained in a derivation as defined in Lemma 2. If C is a closure in N_i , then the number of literals in C is at most polynomial in $|KB|$, for unary coding of numbers in KB input. Furthermore, $|N_i|$ is at most exponential in $|KB|$, for unary coding of numbers in KB input.*

Proof. By Lemma 2, N_i can contain only \mathcal{ALCHIQ}^- -closures. Since redundancy elimination is applied eagerly, N_i cannot contain closures with duplicate literals or closures identical up to variable renaming. Let r denote the number

of role predicate names, c the number of concept predicate names, i the number of individual names and f the number of function symbols occurring in the signature of $\Xi(KB)$. Then r and i are obviously linear in $|KB|$. Furthermore, c is also linear in $|KB|$ since the number of new concept names introduced during preprocessing is bounded by the number of subconcepts of each concept, which is linear in $|KB|$. The number f is bounded by the sum of all numbers n in $\geq n R.C$ or $\leq n R.C$ plus one for each $\exists R.C$ and $\forall R.C$ in KB . Since unary coding of numbers is employed, f is linear in $|KB|$. Let n denote the maximal number occurring in number restrictions in KB . For unary coding, n is linear in $|KB|$.

No inference from \mathcal{BS}_{DL} increases the number of variables in a closure, so the number of variables is bounded by n . Then, we have at most $(f + 1)^2(n + i)$ terms of depth at most 2, which by counting in all possible markings, yield at most $t = 2(f + 1)^2(n + i)$ terms in a closure. This yields at most $ct + rt^2$ atoms, which, together with the equality literals, and allowing each atom to occur negatively, gives at most $\ell = 2(ct + (r + 1)t^2)$ literals in a closure, which is obviously polynomial in $|KB|$ for unary coding of numbers.

Each closure can contain an arbitrary subset of these literals, so the total number closures is bounded by 2^ℓ , so the number of closures unique up to variable renaming is exponential in $|KB|$ for unary coding of numbers. \square

Using binary coding of numbers, it is possible to encode the number n in $\log_2 n$ bits. In this case, f and n are exponential in $|KB|$, thus giving an exponential bound on the number of literals in a closure, and a double exponential bound on the number of closures.

Theorem 2. *For an \mathcal{ALCHIQ}^- knowledge base KB , \mathcal{BS}_{DL} decides satisfiability of KB and runs in time exponential in the size of the input for unary coding of numbers.*

Proof. The translation of KB to $\Xi(KB)$ can be performed in time polynomial in the size of KB and contains only \mathcal{ALCHIQ}^- -closures by Lemma 1. Let c denote the maximal number of closures occurring in the closure set in a derivation as specified in Lemma 2, and let l denote the maximal number of literals in a closure. By Lemma 4, c is exponential, and l polynomial in $|KB|$, for unary coding of numbers. Hence, ordering constraints can be checked in polynomial time. In the worst case, a single subsumption check requires exponential time in the number of literals of the clauses involved (Gottlob & Leitsch 1985). Furthermore, a subsumption check is performed at most for each pair of closures. Hence, subsumption checking takes exponential time in $|KB|$. Each closure can potentially participate in an inference with each other closure, resulting in c^2 combinations. Furthermore, an inference rule can be applied to any pair of literals, resulting in l^2 combinations. Finally, any of the 5 inference rules may be applied. Hence, the number of applications of inference rules of \mathcal{BS}_{DL} is bounded by $5c^2l^2$, which is exponential in $|KB|$, for unary coding of numbers. Now it is obvious that, after at most an exponential number of steps, the set of closures will be saturated, and the procedure will terminate. Since \mathcal{BS}_{DL} is sound and

complete with eager application of redundancy elimination rules, the claim of the theorem follows. \square

Reducing \mathcal{ALCHIQ}^- to Disjunctive Datalog

Based on the decision procedure from the previous section, we show now how to reduce an \mathcal{ALCHIQ}^- knowledge base KB to a disjunctive datalog program. Marking information is not relevant for the reduction to datalog, so in this section we consider any closure $C \cdot \sigma$ equivalent to the clause $C\sigma$.

Eliminating Function Symbols. For some \mathcal{ALCHIQ}^- knowledge base KB , let $\Gamma_{\mathcal{TR}} = \Xi(KB_{\mathcal{T}} \cup KB_{\mathcal{R}})$. Let $\text{Sat}_{\mathcal{R}}(\Gamma_{\mathcal{TR}})$ denote the *relevant set of saturated clauses*, that is, clauses of type 1, 2, 5, 7 obtained by saturating $\Gamma_{\mathcal{TR}}$ using \mathcal{BS}_{DL} with eager application of redundancy elimination rules. Finally, let $\Gamma = \text{Sat}_{\mathcal{R}}(\Gamma_{\mathcal{TR}}) \cup \Xi(KB_{\mathcal{A}})$. Intuitively, $\text{Sat}(\Gamma_{\mathcal{TR}})$ contains all non-redundant clauses following from the TBox and RBox. From this clause set, any further inference involved in deriving the empty clause will involve an ABox clause, which cannot participate in an inference with a clause of type 3, 4, 6 or 8. Hence, we may safely delete these clauses and consider only the $\text{Sat}_{\mathcal{R}}(\Gamma_{\mathcal{TR}})$ subset.

Lemma 5. *KB is unsatisfiable iff Γ is unsatisfiable.*

Proof. KB is unsatisfiable iff the set of clauses derived by the saturation of $\Xi(KB)$ by \mathcal{BS}_{DL} contains the empty clause. Since choosing the premises of each inference rule is don't-care non-deterministic, we may perform all non-redundant inferences among clauses from $\Gamma_{\mathcal{TR}}$ first. Let us denote the resulting set of intermediate clauses with $N_i = \text{Sat}(\Gamma_{\mathcal{TR}}) \cup \Xi(KB_{\mathcal{A}})$. If N_i contains the empty clause, Γ contains it by definition as well (the empty clause is of type 5), and the claim of the lemma follows. Otherwise, we continue with saturation of N_i . Obviously, each N_j , $j > i$, in the derivation, will be obtained from N_{j-1} by applying an inference rule involving at least one clause not in N_i , which can only be a clause of type 7 where u is a constant, or a clause of type 9. By Lemma 3, we may safely consider only derivations where the variables y_k in a clause are assigned terms in the decreasing order. Hence, N_j may not be obtained by resolving a clause of type 7 where u is a constant with a clause of type 3: this would assign y_k to $f(u)$, which is obviously larger than the constant that was assigned to some $y_{k'}$, $k' < k$. Furthermore, from the proof of Lemma 2, one may see that a clause of type 7 where u is a constant cannot participate in a resolution with a clause of type 3, since the unifier never exists. The same lemma shows that any other inferences with clauses of types 3, 4, 6 or 8 are either not possible, or are redundant. Therefore, we may conclude that no clause of type 3, 4, 6 or 8 from N_i participates in deriving N_j , $j > i$. Hence, N_i may safely be replaced by Γ . Any set of clauses N_j , $j > i$, which can be obtained by saturation from $\Xi(KB)$ may be obtained by saturation from Γ as well, modulo clauses of type 3, 4, 6 or 8. Hence, the saturation of Γ by \mathcal{BS}_{DL} derives the empty clause iff the saturation of $\Xi(KB)$ by \mathcal{BS}_{DL} derives the empty clause, so the claim of the lemma follows. \square

We now show how to eliminate function symbols from clauses in Γ . Intuitively, the idea is to replace each ground functional term $f(a)$ with a new constant, denoted as a_f . For each function symbol f we introduce a new predicate symbol S_f , containing, for each constant a , a tuple of the form $S_f(a, a_f)$. Thus, S_f contains the f -successor of each constant. Any reference to a term $f(x)$ in some clause is then replaced with a new variable x_f , with the literal $\neg S_f(x, x_f)$ being added to the clause. Thus, for some a , resolving $\neg S_f(x, x_f)$ with $S_f(a, a_f)$ will bind the value of x_f to a_f , which plays the role of $f(a)$. The Herbrand universe of the clause set becomes thus finite, so it can be represented as a finite relation HU containing all constants a and a_f , and is used to bind unsafe variables.

In order to formalize this process, we first define an operator λ which eliminates functional terms and binds all unsafe variables in a clause.

Definition 7. Let KB be an \mathcal{ALCHIQ}^- knowledge base. For some ground functional term $f(a)$, let $\lambda(f(a))$ denote a globally unique constant a_f , not occurring in KB ¹. For an \mathcal{ALCHIQ}^- -clause C , we define $\lambda(C)$ as follows:

1. For each term of the form $f(x)$ in C , introduce a fresh variable x_f not occurring in C . Replace each occurrence of $f(x)$ with x_f .
2. Replace each ground functional term $f(a)$ with $\lambda(f(a))$.
3. For each variable x_f introduced in the first step, append the literal $\neg S_f(x, x_f)$.
4. If after steps 1–3 some variable x occurs in a positive literal but not in a negative literal, append the literal $\neg HU(x)$.

If p is a position in a clause C , let $\lambda(p)$ denote the corresponding position in $\lambda(C)$. Let λ^- denote the inverse of λ (i.e. $\lambda(\lambda^-(C)) \equiv C$ for any clause C).

Let $\text{FF}(KB) = \text{FF}_\lambda(KB) \cup \text{FF}_{Succ}(KB) \cup \text{FF}_{HU}(KB) \cup \Xi(KB_A)$ denote the function-free version of $\Xi(KB)$, where FF_λ , FF_{Succ} and FF_{HU} are defined as follows, where a and f range over all constant and function symbols in $\Xi(KB)$:

$$\begin{aligned} \text{FF}_\lambda(KB) &= \bigcup_{C \in \text{Sat}_R(\Gamma_{TR})} \lambda(C) \\ \text{FF}_{Succ}(KB) &= \bigcup S_f(a, \lambda(f(a))) \\ \text{FF}_{HU}(KB) &= \bigcup HU(a) \cup \bigcup HU(\lambda(f(a))) \end{aligned}$$

We now show that KB and $\text{FF}(KB)$ are equi-satisfiable.

Lemma 6. KB is unsatisfiable iff $\text{FF}(KB)$ is unsatisfiable.

Proof. We show that Γ and $\text{FF}(KB)$ are equi-satisfiable. Since KB and Γ are equi-satisfiable by Lemma 5, the claim of the lemma follows.

(\Leftarrow) If $\text{FF}(KB)$ is unsatisfiable, since hyperresolution with superposition and splitting is sound and complete (Bachmair & Ganzinger 1994), a derivation of an empty clause exists. We now show that each such a derivation can be reduced to a derivation of the empty clause in Γ by sound inference

¹Globally unique means that, for some f and a , the constant a_f is always the one and the same.

rules, in particular, hyperresolution, paramodulation, instantiation and splitting. In $\text{FF}(KB)$, all clauses are safe, so electrons are always positive ground clauses, and each hyperresolvent is a positive ground clause. Furthermore, since superposition into variables is not necessary for completeness, superposition-related inferences are necessary only among ground clauses. Finally, splitting ground clauses simplifies the proof, since all ground clauses on each branch are unit clauses.

Let B be a branch $\text{FF}(KB) = N_0, \dots, N_n$ of a derivation by hyperresolution with superposition and eager splitting from $\text{FF}(KB)$. We show now by induction on n that, for any branch B , there exists a corresponding branch B' in a derivation from Γ by sound inference steps, and a set of clauses N'_m on B' such that: (*) if C is some clause in N_n not of the form $S_f(u, v)$ or $HU(u)$, then N'_m contains the counterpart clause of C , equal to $\lambda^-(C)$. The induction base $n = 0$ is obvious, as $\text{FF}(KB)$ and Γ contain only one branch, on which, other than $S_f(u, v)$ or $HU(u)$, all ground clauses are ABox clauses. Now assume that the proposition (*) holds for some n and consider all possibilities for the inference of a clause C from clauses in N_n , forming N_{n+1} :

- Superposition into a literal $HU(u)$ is redundant, since the predicate HU is instantiated for each constant occurring in $\text{FF}(KB)$, so the conclusion already appears on the branch.
- Assume that the inference is a superposition from $s \approx t$ into the ground unit clause L . If L is of the form $S_f(u, v)$, then the proposition obviously holds. Otherwise, clauses $s \approx t$ and L are derived in at most n steps on B , so by induction assumption counterpart clauses $\lambda^-(s \approx t)$ and $\lambda^-(L)$ are derivable in B' . Thus, superposition can be performed on these clauses in B' , so the proposition holds.
- Reflexivity resolution can only be performed on some clause $u \approx u$ in B . By induction hypothesis $\lambda^-(u \approx u)$ is then derivable in B' , and reflexivity resolution can be applied there, so the proposition holds.
- Equality factoring is not applicable to B , since all positive clauses in B are ground unit clauses.
- Assume that the inference is a hyperresolution inference with nucleus C , the set of positive ground electrons E_1, \dots, E_k , and the unifier σ , resulting in the hyperresolvent H . We construct the substitution σ' as follows: for each variable $x \in \text{dom}(\sigma)$ not of the form x_f , we include a mapping $x \mapsto \lambda^-(x\sigma)$. Let us now perform on B' an instantiation step $C' = (\lambda^-(C))\sigma'$. Obviously, $\lambda^-(C\sigma)$ and C' may differ only at a position p in C , at which a variable of the form x_f occurs. Let us denote with p' the position $\lambda^-(p)$ in C' . Furthermore, the term at p' in $\lambda^-(C)$ is $f(x)$, so with p'_x we denote the position of the inner x in $f(x)$. In the hyperresolution inference generating H , the variable x_f is instantiated by resolving $\neg S_f(x, x_f)$ with some ground literal $S_f(u, v)$. Hence, $C\sigma$ contains at p the term v , whereas C' contains at p' the term $f(u)$, and $\lambda^-(v) \neq f(u)$. We show now how to eliminate all such discrepancies in B' . Observe that the literal $S_f(u, v)$ is on B obtained from some

$R = S_f(a, a_f)$ by n or less superposition inference steps. Let us denote by Δ_1 (Δ_2) the sequence of ground unit equalities applied to the first (second) argument of R . All $s_i \approx t_i$ from Δ_1 or Δ_2 are derivable in n steps or less on B , so corresponding equalities $\lambda^-(s_i \approx t_i)$ are derivable on B' by induction hypothesis. Let us denote these corresponding sequences with Δ'_1 and Δ'_2 . We may now perform superposition with equalities from Δ'_1 to C' at p'_x in the reverse order. After this, p'_x will contain the constant a , and p' will contain the term $f(a)$. Hence, we may now apply superposition with equalities from Δ'_2 at p' in the original order. After this is done, each position p' will contain the term $\lambda^-(v)$. Let us denote with C'' the result of removing discrepancies at all positions. Obviously, $C'' = \lambda^-(C\sigma)$. All electrons E_i are derivable in n steps or less on B , so if E_i is not of the form $S_f(u, v)$ or $HU(u)$, $\lambda^-(E_i)$ is derivable on B' . We may now hyper-resolve these electrons with C'' to obtain H' . Obviously, $H' = \lambda^-(H)$, so the proposition holds.

- If some ground clause C of length k causes the branch B to be split into k sub-branches, then $\lambda^-(C)$ is also of length k and B' can be split into k sub-branches, where each of them satisfies (*), so the proposition holds.

Hence, if there is a derivation of the empty clause on all branches from $\text{FF}(KB)$, then there is a derivation of the empty clause on all branches from Γ as well.

(\Rightarrow) If Γ is unsatisfiable, since \mathcal{BS}_{DL} is sound and complete, a derivation of an empty clause exists. We now show that each such derivation can be reduced to a derivation of the empty clause in $\text{FF}(KB)$ by sound inference rules.

Let B' be a derivation $\Gamma = N'_0, \dots, N'_n$ by \mathcal{BS}_{DL} . We show by induction on n that there exists a corresponding derivation B of the form $\text{FF}(KB) = N_0, \dots, N_m$ by sound inference steps, such that: (***) if C' is some clause in N'_n , then N_m contains the counterpart clause $C = \lambda(C')$. The induction base $n = 0$ is trivial. Assume now that (***) holds for some n and consider possible inferences deriving $N'_{n+1} = N'_n \cup \{C'\}$, where the clause C' is derived from premises P'_1 and P'_2 in N'_n . By induction hypothesis, we know that there is a derivation B from $\text{FF}(KB)$ with a clause set N_m containing the counterpart clauses of the premises P'_1 and P'_2 , denoted with P_1 and P_2 , respectively. We now consider each possible inference that might have lead to the derivation of C' and show how to construct a derivation of $C = \lambda(C')$ from N_m .

Assume that the inference is by ordered resolution on literals $L'_1 \in P'_1$ and $L'_2 \in P'_2$. Then resolution may be applied on corresponding literals L_1 and L_2 of P_1 and P_2 , respectively, resulting in a clause D . Unification of a non-ground functional term $f(x)$ with some other term or variable in L'_1 and L'_2 corresponds to the unification of x_f with some other term or variable in L_1 and L_2 . The differences between $\lambda(C')$ and D may have the following causes:

- C' may have some term $f(a)$ appearing in C' at position p , while D contains x_f at $\lambda(p)$. However, D then con-

tains the literal $\neg S_f(a, x_f)$, which can be resolved with $S(a, a_f)$, to produce a_f at position $\lambda(p)$.

- $\lambda(C')$ and D may differ in some literal of the form $\neg HU(u)$. Since, for any constant u , any set of clauses on B contains $HU(u)$, this discrepancy can easily be removed by resolving C with $HU(u)$.

By successively removing differences between D and $\lambda(C')$, we eventually obtain a clause C such that $C = \lambda(C')$.

If the inference is by equality factoring or reflexivity resolution, then the premise P'_1 is ground and the inference may be applied to P_1 in the same way.

Assume the inference is by positive or negative basic superposition. If both P_1 and P_2 are ground, since superposition into Skolem function symbols is not needed, superposition can be applied to P_1 and P_2 in the same way. Otherwise, P_1 is a clause of type 5. Let superposition be performed at position p into a term of the form $f(x)$ with the term in P_2 being of the form $f(a)$, with unifier $\{x \mapsto a\}$. This inference can be simulated in N_m as follows: P_1 must contain a literal $\neg S_f(x, x_f)$ and the variable x_f must occur at position $\lambda(p)$. One can first resolve P_1 with $S_f(a, a_f)$, which will produce a_f at position $\lambda(p)$. Now one may perform superposition with P_2 at $\lambda(p)$ to obtain the clause C . Since P_1 contains x_f , is it safe and does not contain any $\neg HU(x)$ literals, so $C = \lambda(C')$.

Hence, if there is a derivation of the empty clause from Γ , then there is a derivation of the empty clause from $\text{FF}(KB)$ as well. \square

The result above means that $KB \models \alpha$ iff $\text{FF}(KB) \models \alpha$, where α may be of the form $(\neg)A(a)$ or $(\neg)R(a, b)$, where A is an atomic concept. The proof also reveals the fact that, in checking satisfiability of $\text{FF}(KB)$, it is not necessary to perform superposition into literals $HU(a)$.

In case the knowledge base uses only constructs from the \mathcal{ALCHT} subset, further optimizations are possible, since $\Xi(KB)$ then does not contain equalities. The proof of Lemma 2 implies that clauses of type 5 containing a functional term cannot participate in any inference with clauses of type 9: superposition into $f(x)$ is not possible, so no ground literal containing a functional term may be generated. In this case, $\text{Sat}_R(\Gamma_{\mathcal{TR}})$ should contain only function-free clauses from the saturated set. Also, $\text{FF}(KB)$ should contain only $HU(a)$ for each constant a .

Removing Irrelevant Clauses. The saturation of $\Gamma_{\mathcal{TR}}$ derives new clauses which enable the reduction to $\text{FF}(KB)$. However, the same process introduces lots of clauses which are not necessary. Consider, for example, the knowledge base $KB = \{A \sqsubseteq C, C \sqsubseteq B\}$. If the predicate ordering is $C \succ B \succ A$, then the saturation process will derive the clause $\neg A(x) \vee B(x)$, which is not necessary: all ground consequences of this clause may be obtained by combining ground consequences of the first two. Hence, we now present an optimization, by which we reduce the number of clauses in the resulting disjunctive datalog program.

Definition 8. Let $C \in \text{FF}(KB)$ be a clause such that $\lambda^-(C)$ was derived in the saturation of $\Gamma_{\mathcal{TR}}$ from premises P_i , $1 \leq i \leq k$, by an inference with a substitution σ . Then C is irrelevant in $\text{FF}(KB)$ if, for each premise P_i , $\lambda(P_i)$ is defined, $\lambda(P_i) \in \text{FF}(KB)$, and each variable occurring in $\lambda(P_i\sigma)$ occurs in C . A clause C is relevant iff it is not irrelevant. Finally, we use $\text{FF}_R(KB)$ to set of all clauses relevant in $\text{FF}(KB)$.

Removing irrelevant clauses preserves satisfiability, as demonstrated by the following lemma.

Lemma 7. $\text{FF}_R(KB)$ is unsatisfiable iff $\text{FF}(KB)$ is unsatisfiable.

Proof. Let C be an irrelevant clause in $\text{FF}(KB)$, where $\lambda^-(C)$ is derived in the saturation of $\Gamma_{\mathcal{TR}}$ from premises P_i by an inference rule ξ with a substitution σ . Let N be a (not necessarily proper) subset of $\text{FF}(KB)$, such that $C \in N$ and $\lambda(P_i) \in N$, $1 \leq i \leq k$. We now demonstrate the following property (***) : N is unsatisfiable iff $N \setminus \{C\}$ is unsatisfiable. The (\Leftarrow) direction is trivial, since $N \setminus \{C\} \subset N$.

For the (\Rightarrow) direction, by Herbrand's theorem, N is unsatisfiable iff some finite set M of ground instances of N is unsatisfiable. For such M , we construct the set of ground clauses M' in the following way, where $\lambda(\sigma)$ is the substitution obtained from σ by changing each $x \mapsto t$ into $x \mapsto \lambda(t)$:

- For each $D \in M$ such that D is not a ground instance of C , let $D \in M'$.
- For each $D \in M$ such that D is a ground instance of C with substitution τ , let $\lambda(P_i)\lambda(\sigma)\tau \in M'$, $1 \leq i \leq k$.

Let τ be a ground substitution for C and $D = C\tau$. Since P_i can be clauses of type 1 – 5, and σ is the most general unifier, it can contain only mappings of the form $x \mapsto c$, $x \mapsto x'$ or $x \mapsto f(x')$. Hence, the set of variables in $\lambda(P_i\sigma)$ and $\lambda(P_i)\lambda(\sigma)$ coincide, and since τ instantiates all variables from $\lambda(P_i\sigma)$, the clauses in M' are indeed ground instances of $N \setminus \{C\}$. Furthermore, it is easy to see that $\lambda(P_i)\lambda(\sigma)\tau \subseteq \lambda(P_i\sigma)\tau$. If the inclusion is strict, this is due to literals of the form $\neg S_f(a, b)$ in the latter clause which are not in the first one because σ instantiates some variable from P_i to a functional term $f(x')$ originating from some premise P_j . But then $\lambda(P_j)$ contains the literal $\neg S_f(x', x'_f)$, so $\lambda(P_j)\lambda(\sigma)\tau$ contains $\neg S_f(a, b)$. Therefore, all $\lambda(P_i)\lambda(\sigma)\tau$ can participate in an ground inference corresponding to ξ deriving D , so if M is unsatisfiable, M' is unsatisfiable as well. Since M' is an unsatisfiable set of ground instances of $N \setminus \{C\}$, $N \setminus \{C\}$ is unsatisfiable by Herbrand's theorem.

Let *derives* be a binary relation on clauses in $\text{FF}(KB)$, such that C_1 *derives* C_2 if $\lambda^-(C_1)$ was used as a premise for deriving $\lambda^-(C_2)$ in the saturation of $\Gamma_{\mathcal{TR}}$. Obviously, *derives* is a directed acyclic graph, so it can be topologically sorted into a sequence C_1, \dots, C_n , such that for each $1 \leq i < j \leq n$, no C_i *derives* some C_j (i.e. each clause has a smaller index than the clauses it was derived from). Consider now a sequence of clause sets $N_0 = \text{FF}_R(KB)$, N_1, \dots, N_n , where $N_i = N_{i-1}$ if C_i is relevant in $\text{FF}(KB)$, and $N_i = N_{i-1} \setminus \{C_i\}$ if C_i is irrelevant in

$\text{FF}(KB)$, $1 \leq i \leq n$. By induction on n , N_n is unsatisfiable iff $\text{FF}(KB)$ is unsatisfiable: if C_i is irrelevant, since all premises deriving $\lambda^-(C_i)$ are in N_i , the conditions of (***) are satisfied. Furthermore, all irrelevant clauses are eliminated in N_n . Hence, $N_n = \text{FF}_R(KB)$, and the claim of the lemma follows. \square

Reduction to Disjunctive Datalog. Computing the reduction of an \mathcal{ALCHIQ}^- knowledge base KB to disjunctive datalog is now easy.

Definition 9. Reduction of KB to a disjunctive datalog program $\text{DD}(KB)$ is obtained by simply rewriting each clause $A_1 \vee \dots \vee A_n \vee \neg B_1 \vee \dots \vee \neg B_m$ in $\text{FF}_R(KB)$ as the rule $A_1 \vee \dots \vee A_n \leftarrow B_1, \dots, B_m$.

Theorem 3. Let KB be an \mathcal{ALCHIQ}^- knowledge base and let $\text{DD}(KB)$ be its reduction to disjunctive datalog². Then the following claims hold:

1. KB is unsatisfiable iff $\text{DD}(KB)$ is unsatisfiable.
2. $KB \models \alpha$ iff $\text{DD}(KB) \models_c \alpha$, where α is of the form $A(a)$ or $R(a, b)$ and A is an atomic concept.
3. $KB \models C(a)$ with C being a non-atomic concept iff $\text{DD}(KB \cup \{C \sqsubseteq Q\}) \models_c Q(a)$.
4. The number of rules in $\text{DD}(KB)$ is at most exponential, the number of literals in each rule is at most polynomial, and $\text{DD}(KB)$ can be computed in exponential time in $|KB|$, for unary coding of numbers in the input.

Proof. The first claim is an obvious consequence of Lemma 7. The second claim follows from the first one, since $\text{DD}(KB \cup \{\neg\alpha\}) = \text{DD}(KB) \cup \{\neg\alpha\}$ is unsatisfiable iff $\text{DD}(KB) \models_c \alpha$. Also, $KB \models C(a)$ iff $KB \cup \neg C(a)$ is unsatisfiable, which is the case iff $KB \cup \{\neg Q(a), \neg Q \sqsubseteq \neg C\} = KB \cup \{\neg Q(a), C \sqsubseteq Q\}$ is unsatisfiable. Now the third claim follows from the second one, and the fact that Q is atomic.

By Lemma 4, $|\text{Sat}(\Gamma_{\mathcal{TR}})|$ is at most exponential in $|KB|$, and, for each clause C in it, the number of literals is at most polynomial in $|KB|$. It is easy to see that the application of λ to C can be performed in time polynomial in the number of terms and literals in C . The number of constants a_f added to $\text{DD}(KB)$ is equal to $i \cdot f$, where i is the number of individuals, and f the number of function symbols. By Lemma 4, if numbers are unarily coded, both i and f are polynomial in $|KB|$, so the number of constants a_f is also polynomial in $|KB|$. By Theorem 2, $\text{Sat}(\Gamma_{\mathcal{TR}})$ can be computed in time at most exponential in $|KB|$, so the fourth claim follows. \square

In (Hustadt, Motik, & Sattler 2003), we show that all certain answers of $\text{DD}(KB)$ of the form $Q(a)$, where the predicate Q does not occur in the body of some rule in $\text{DD}(KB)$, can be computed by saturating $\text{DD}(KB)$ by hyperresolution and basic superposition under any ordering where all literals with the predicate Q are smallest. Furthermore, we have shown that saturation can be performed in exponential time,

²With $P \models_c A$ we denote the cautious entailment of A from P , where A must be contained in every minimal model of P .

