Reasoning in Description Logics by a Reduction to Disjunctive Datalog

Ullrich Hustadt
Department of Computer Science, University of Liverpool, Liverpool, UK

Boris Motik
Department of Computer Science, University of Manchester, Manchester, UK

Ulrike Sattler
Department of Computer Science, University of Manchester, Manchester, UK

Abstract. As applications of description logics proliferate, efficient reasoning with knowledge bases containing many assertions becomes ever more important. For such cases, we developed a novel reasoning algorithm that reduces a $\mathcal{SHIQ}$ knowledge base to a disjunctive datalog program while preserving the set of ground consequences. Queries can then be answered in the resulting program while reusing existing and practically proven optimization techniques of deductive databases, such as join-order optimizations or magic sets. Moreover, we use our algorithm to derive precise data complexity bounds: we show that $\mathcal{SHIQ}$ is data complete for NP, and we identify an expressive fragment of $\mathcal{SHIQ}$ with polynomial data complexity.

Keywords: Description Logics, Disjunctive Datalog, Data Complexity

1. Introduction

In recent years, description logics (DLs) have found their application in various fields of computer science, such as data integration, knowledge representation, and ontology modeling for the Semantic Web [10][3, Part III]. A DL knowledge base typically consists of two parts. The *terminological* part of a knowledge base, called TBox, can be thought of as the "schema" since it contains concept definitions and background knowledge. A central reasoning task for TBoxes is the computation of the subsumption hierarchy, which is based on deciding entailment of implications between formulae w.r.t. a background theory. The *assertional* part of a knowledge base, called ABox, can be thought of as the "data" since it contains ground facts. The main reasoning task for ABoxes is query answering, which, in its simplest form, amounts to retrieving all instances of a certain concept.

$\mathcal{SHIQ}$ [21] is a very expressive DL that provides the logical underpinning for the OWL-Lite and OWL-DL variants of the Web Ontology Language (OWL) [20]. Several practical reasoners for logics around $\mathcal{SHIQ}$ have been built [34, 17, 39] and applied to practical problems. Experience shows that these systems perform well when computing

the subsumption hierarchy: they use practical, optimized tableau-based algorithms [21], which perform much better on practical problems than their worst-case computational complexity suggests. New applications, such as metadata management in the Semantic Web, require also efficient query answering over large ABoxes. So far, query answering was implemented by reducing the problem to ABox consistency checking, which can then be solved using tableau algorithms [18].

In this paper, we propose a completely different and novel approach to this complex reasoning problem. It is based on reusing ideas from deductive databases—an extension of the relational database model with deductive features, usually in the form of (recursive) rules [1]. Reasoning algorithms and optimization techniques for deductive databases were specifically designed to handle large data quantities. In order to apply them to DL reasoning, we developed a novel algorithm that reduces a $\mathcal{SHIQ}$ knowledge base to a disjunctive datalog program. Since the reduction preserves the set of entailed ground facts, queries can be answered in the resulting program while reusing all existing optimization techniques.

Our reduction to datalog does not mean that we suggest employing nonmonotonic negation or minimal model reasoning. Rather, we consider disjunctive datalog useful because it allows the application of the join order optimization, which, based on the database statistics, chooses the data access path promising the least cost [1]. Moreover, our algorithm allows us to apply the magic sets transformation [7]. Roughly speaking, this transformation modifies the datalog program such that bottom-up evaluation of the transformed program simulates top-down parameter passing in the original program. In this way, only the facts directly relevant to the query answer are considered. This technique was crucial in providing efficient reasoning in deductive databases containing large data quantities. The magic sets transformation for disjunctive programs has been presented in [12], along with empirical evidence of its usefulness. We are unaware of an approach that would allow us to apply the magic sets transformation directly in the resolution setting, without a prior transformation into disjunctive datalog.

As an added benefit, our algorithm separates TBox from ABox reasoning. Thus, the TBox inferences are not being repeated for different individuals during query answering.

Our algorithm runs in worst-case exponential time. Furthermore, query answering in the resulting program can be performed in time exponential in the size of the input knowledge base, so reasoning in $\mathcal{SHIQ}$ using our algorithms has optimal worst-case complexity, assuming unary coding of numbers. The latter assumption is quite common

in practical implementations of description logics, even though $\mathcal{SHIQ}$ is ExpTime-complete regardless of the coding of numbers [38].

As a side-effect of our reduction, we obtain novel *data complexity* results. Assuming that the ABox contains only possibly negated atomic concepts, we show that checking KB satisfiability is NP-complete and query answering is co-NP-complete in the size of the ABox. This is still intractable, so we identify Horn-$\mathcal{SHIQ}$, a fragment of $\mathcal{SHIQ}$ analogous to the Horn fragment of first-order logic. Namely, Horn-$\mathcal{SHIQ}$ provides existential and universal quantifiers, but does not provide means to express disjunctive information. We show that the basic reasoning problems for Horn-$\mathcal{SHIQ}$ are P-complete in the size of the ABox.

The algorithms from this paper are implemented in KAON2[1]—a new DL reasoner. Our performance comparison of KAON2 with other state-of-the-art reasoners shows that our algorithm indeed leads to performance improvements in query answering of one or more orders of magnitude on large knowledge bases [29].

This paper contains an extended version of the results that have been published in [22, 23, 24, 28].

2. Preliminaries

2.1. Description Logic $\mathcal{SHIQ}$

The syntax of $\mathcal{SHIQ}$ is given by the following definition [21].

DEFINITION 1. *For a set of* role names N_R*, the set of* roles *is the set* $N_R \cup \{R^- \mid R \in N_R\}$*. For* $R \in N_R$*, let* $\mathsf{Inv}(R) = R^-$ *and* $\mathsf{Inv}(R^-) = R$*. An* RBox $KB_{\mathcal{R}}$ *is a finite set of* transitivity axioms $\mathsf{Trans}(R)$ *and* role inclusion axioms $R \sqsubseteq S$ *for* R *and* S *roles. Let* $\sqsubseteq^*$ *be the reflexive-transitive closure of* $\{R \sqsubseteq S, \mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in KB_{\mathcal{R}}\}$*. A role* R *is* transitive *in* $KB_{\mathcal{R}}$ *if a role* S *exists such that* $S \sqsubseteq^* R$*,* $R \sqsubseteq^* S$*, and either* $\mathsf{Trans}(S) \in \mathcal{R}$ *or* $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{R}$*;* R *is* simple *if no transitive role* S *exists with* $S \sqsubseteq^* R$*; and* R *is* complex *if it is not simple.*

Let N_C *be a set of* atomic *concepts. The set of* concepts *over* N_C *and* N_R *is the smallest set such that* $\top$ *(*top concept*) and* $\bot$ *(*bottom concept*) are concepts, each atomic concept* $A \in N_C$ *is a concept, and, if* C *and* D *are concepts,* R *is a role,* S *is a simple role, and* n *is a nonnegative integer, then* $\neg C$ *(*concept complement*),* $C \sqcap D$ *(*concept intersection*),* $C \sqcup D$ *(*concept union*),* $\exists R.C$ *(*existential restriction*),* $\forall R.C$ *(*universal restriction*),* $\leq n\, S.C$ *(*at-most qualified number restriction*), and* $\geq n\, S.C$ *(*at-least qualified number restriction*) are concepts.* Literal concepts *are possibly negated atomic concepts.*

[1] http://kaon2.semanticweb.org/

A TBox $KB_{\mathcal{T}}$ *over* N_C *and* $KB_{\mathcal{R}}$ *is a finite set of* concept inclusion axioms $C \sqsubseteq D$*, for concepts* C *and* D*.*

Let N_I *be a set of* individuals*. An* ABox $KB_{\mathcal{A}}$ *is a set of concept and role* assertions $C(a)$*,* $R(a,b)$*,* $\neg S(a,b)$*, and* (in)equality axioms $a \approx b$ *and* $a \not\approx b$*, for* C *a concept,* R *a role,* S *a simple role, and* a *and* b *individuals. An ABox is* extensionally reduced *if its assertions contain only literal concepts.*

A $\mathcal{SHIQ}$ *knowledge base KB is a triple* $(KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$*. With* $|KB|$ *we denote the size of KB with numbers coded in unary—that is,* $|\leq n\, R.C| = |\geq n\, R.C| = |C| + n + 1$*.*

Each KB can be made extensionally reduced as follows: for each axiom $C(a)$ where C is not a literal concept, introduce a new atomic concept A_C, add the axiom $A_C \sqsubseteq C$ to the TBox, and replace $C(a)$ with $A_C(a)$. This transformation is obviously polynomial in $|KB|$.

For the semantics, we translate DL knowledge bases into a first-order formula. Such a semantics is known to be equivalent to the more commonly used direct model-theoretic semantics [8]. The translation of number restrictions uses counting quantifiers $\exists^{\geq n}$ and $\exists^{\leq n}$. It is well known that these can be represented in first-order logic by means of standard quantifiers and equality as follows, for $\mathbf{y}$ a vector of variables:

$$\exists^{\geq n} x : \varphi(x,\mathbf{y}) = \exists x_1, \ldots, x_n : \left[\bigwedge_{1 \leq i \leq n} \varphi(x_i, \mathbf{y}) \wedge \bigwedge_{1 \leq i < j \leq n} x_i \not\approx x_j \right]$$

$$\exists^{\leq n} x : \varphi(x,\mathbf{y}) = \forall x_1, \ldots, x_{n+1} : \left[\bigwedge_{1 \leq i \leq n+1} \varphi(x_i, \mathbf{y}) \rightarrow \bigvee_{1 \leq i < j \leq n+1} x_i \approx x_j \right]$$

DEFINITION 2. *The semantics of a* $\mathcal{SHIQ}$ *knowledge base KB is defined by transforming KB into the formula* $\pi(KB)$ *of first-order logic with counting quantifiers and equality, where* π *is the mapping operator shown in Table I. KB is* satisfiable *if and only if* $\pi(KB)$ *is satisfiable.*

Other interesting inference problems can be reduced to satisfiability using well-known transformations [3, Chapter 2]. The DL $\mathcal{ALCHIQ}$ is obtained from $\mathcal{SHIQ}$ by prohibiting transitivity axioms, and the DL $\mathcal{ALC}$ is obtained from $\mathcal{ALCHIQ}$ by prohibiting inverse roles, qualified number restrictions, and role inclusion axioms. A logic $\mathcal{L}$ is a *fragment* of a logic $\mathcal{L}'$ if each axiom of $\mathcal{L}$ is also an axiom of $\mathcal{L}'$ and it is interpreted in both logics in the same way. A logic $\mathcal{L}$ is *between* logics $\mathcal{L}_1$ and $\mathcal{L}_2$ if $\mathcal{L}_1$ is a fragment of $\mathcal{L}$, and $\mathcal{L}$ is a fragment of $\mathcal{L}_2$.

A *position* p is a finite sequence of integers; the empty position is denoted with ϵ. A position p is *below* a position q if q is a proper prefix of p. For a concept α, the *subterm at position* p, written $\alpha|_p$, is

Table I. Translation of $\mathcal{SHIQ}$ into FOL

Translating Concepts to FOL	
$\pi_x(\top) = \top$	$\pi_y(\top) = \top$
$\pi_x(\bot) = \bot$	$\pi_y(\bot) = \bot$
$\pi_x(A) = A(x)$	$\pi_y(A) = A(y)$
$\pi_x(\neg C) = \neg\pi_x(C)$	$\pi_y(\neg C) = \neg\pi_y(C)$
$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$	$\pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D)$
$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$	$\pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D)$
$\pi_x(\exists R.C) = \exists y : [R(x,y) \wedge \pi_y(C)]$	$\pi_y(\exists R.C) = \exists x : [R(y,x) \wedge \pi_x(C)]$
$\pi_x(\forall R.C) = \forall y : [R(x,y) \rightarrow \pi_y(C)]$	$\pi_y(\forall R.C) = \forall x : [R(y,x) \rightarrow \pi_x(C)]$
$\pi_x(\geq n\, S.C) = \exists^{\geq n} y : [S(x,y) \wedge \pi_y(C)]$	$\pi_y(\geq n\, S.C) = \exists^{\geq n} x : [S(y,x) \wedge \pi_x(C)]$
$\pi_x(\leq n\, S.C) = \exists^{\leq n} y : [S(x,y) \wedge \pi_y(C)]$	$\pi_y(\leq n\, S.C) = \exists^{\leq n} x : [S(y,x) \wedge \pi_x(C)]$
Translating Axioms to FOL	
$\pi(\mathsf{Trans}(R)) = \forall x,y,z : [R(x,y) \wedge R(y,z) \rightarrow R(x,z)]$	
$\pi(R \sqsubseteq S) = \forall x,y : [R(x,y) \rightarrow S(x,y)]$	$\pi(C \sqsubseteq D) = \forall x : [\pi_x(C) \rightarrow \pi_x(D)]$
$\pi(C(a)) = \pi_x(C)\{x \mapsto a\}$	
$\pi(R(a,b)) = R(a,b)$	$\pi(\neg S(a,b)) = \neg S(a,b)$
$\pi(a \approx b) = a \approx b$	$\pi(a \not\approx b) = a \not\approx b$
Mapping KB to FOL	
$\pi(R) = \forall x,y : R(x,y) \leftrightarrow R^-(y,x)$	
$\pi(KB_{\mathcal{R}}) = \bigwedge_{\alpha \in KB_{\mathcal{R}}} \pi(\alpha) \wedge \bigwedge_{R \in N_R} \pi(R)$	$\pi(KB_{\mathcal{T}}) = \bigwedge_{\alpha \in KB_{\mathcal{T}}} \pi(\alpha)$
$\pi(KB) = \pi(KB_{\mathcal{R}}) \wedge \pi(KB_{\mathcal{T}}) \wedge \pi(KB_{\mathcal{A}})$	$\pi(KB_{\mathcal{A}}) = \bigwedge_{\alpha \in KB_{\mathcal{A}}} \pi(\alpha)$

defined as follows: $\alpha|_\epsilon = \alpha$; $(\neg D)|_{1.p} = D|_p$; $(D_1 \circ D_2)|_{i.p} = D_i|_p$ for $\circ \in \{\sqcap, \sqcup\}$ and $i \in \{1,2\}$; $\alpha|_1 = R$ and $\alpha|_{2.p} = D|_p$ for $\alpha = \Diamond R.D$ and $\Diamond \in \{\exists, \forall\}$; and $\alpha|_1 = n$, $\alpha|_2 = R$, and $\alpha|_{3.p} = D|_p$ for $\alpha = \bowtie n\, R.D$ and $\bowtie \in \{\leq, \geq\}$. A *replacement* of a subterm of α at position p with a term β is denoted as $\alpha[\beta]_p$ and is defined as usual. For a concept α and a position p such that $\alpha|_p$ is a concept, the *polarity* of $\alpha|_p$ in α, written $\mathsf{pol}(\alpha, p)$, is defined as follows:

$$
\begin{array}{rcl@{\qquad}rcl}
\mathsf{pol}(C, \epsilon) & = & 1 & \mathsf{pol}(\geq\, n\, R.C, 3.p) & = & \mathsf{pol}(C, p) \\
\mathsf{pol}(\neg C, 1.p) & = & -\mathsf{pol}(C, p) & \mathsf{pol}(\leq\, n\, R.C, 3.p) & = & -\mathsf{pol}(C, p) \\
\mathsf{pol}(C_1 \circ C_2, i.p) & = & \mathsf{pol}(C_i, p) \text{ for } \circ \in \{\sqcap, \sqcup\} \text{ and } i \in \{1,2\} & & & \\
\mathsf{pol}(\Diamond R.C, 2.p) & = & \mathsf{pol}(C, p) \text{ for } \Diamond\ \in \{\exists, \forall\} & & &
\end{array}
$$

2.2. Basic Superposition Calculus

Basic superposition [6, 31] is a clausal calculus optimized for theorem proving with equality. We use standard definitions of terms, atoms, and literals. For convenience, we distinguish constants from function symbols by requiring function symbols to have nonzero arity. We write positive equality literals as $s \approx t$, and negative equality literals as $s \not\approx t$. A clause is a finite multiset of literals; a clause with one literal is called a *unit* clause. As we do for concepts in Section 2.1, we also use positions to describe an "address" of a subterm in a term; $t|_p$ is a subterm of t at position p; and $t[s]_p$ is a replacement of a subterm of t at position p with the term s. We extend these notions to literals and clauses in the obvious way.

It is common practice in equational theorem proving to consider logical theories containing only the equality predicate, as this simplifies the theoretical treatment without loss of generality. A literal $P(t_1, \ldots, t_n)$, where P is not the equality predicate, is encoded as $P(t_1, \ldots, t_n) \approx \mathsf{tt}$, where tt is a new constant. Assuming that P and tt are of sort different from the sort of terms t_i, this encoding preserves satisfiability. Technically speaking, P thus becomes a function symbol; however, when ambiguity does not arise, we call it a predicate symbol. We take $P(t_1, \ldots, t_n)$ to be a syntactic shortcut for $P(t_1, \ldots, t_n) \approx \mathsf{tt}$. Furthermore, we assume the predicate $\approx$ to have built-in symmetry: a literal $s \approx t$ also denotes the literal $t \approx s$, and a literal $s \not\approx t$ also denotes the literal $t \not\approx s$.

The inference rules of basic superposition 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 with $[\cdot]$. A position at or beneath a marked position is called a *substitution position*. For example, 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 basic superposition calculus requires two parameters. The first is an *admissible* ordering $\succ$ on terms—that is, a *reduction ordering* total on ground terms. The second parameter of the calculus is a *selection function*, which selects an arbitrary set of negative literals in a closure.

A term ordering $\succ$ can be extended to an ordering on literals, which we also denote with $\succ$. If $\succ$ is total on nonground terms (as it is the case in this paper), it can be extended to literals by assigning to each literal $L = s \bowtie t$ with $\bowtie \in \{\approx, \not\approx\}$ a complexity measure

$c_L = (\max(s,t), p_L, \min(s,t))$, where p_L is 1 if $\bowtie$ is $\approx$, and 0 otherwise. Then, $L_1 \succ L_2$ iff $c_{L_1} \succ c_{L_2}$, where c_{L_i} are compared lexicographically, with $1 \succ 0$. A literal $L \cdot \theta$ is *maximal* w.r.t. a closure $C \cdot \theta$ if there is no literal $L' \in C$ such that $L'\theta \succ L\theta$; furthermore, $L \cdot \theta$ is *strictly maximal* w.r.t. $C \cdot \theta$ if there is no literal $L' \in C$ such that $L'\theta \succeq L\theta$.

A literal $L \cdot \theta$ is *(strictly) eligible for superposition* in a closure $(C \vee L) \cdot \theta$ if there are no selected literals in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is (strictly) maximal w.r.t. $C \cdot \theta$. A literal $L \cdot \theta$ is *eligible for resolution* in a closure $(C \vee L) \cdot \theta$ if it is selected in $(C \vee L) \cdot \theta$, or if there are no selected literals in $(C \vee L) \cdot \theta$ and $L \cdot \theta$ is maximal w.r.t. $C \cdot \theta$. The basic superposition calculus, $\mathcal{BS}$ for short, consists of the inference rules from Table II. Semantically, all closures are universally quantified, so we can assume each closure to contain a distinct set of variables and all premises to contain the same substitution ρ. Additionally, basic superposition comes with *redundancy elimination rules*, which allow the removal of certain *redundant* closures in a saturation without affecting completeness [6, 31]. For example, a closure $C_1 = A(x)$ *subsumes* a closure $C_2 = A([f(x)]) \vee B(x)$ because, roughly speaking, by instantiating the variable x in C_1 to $f(x)$ we obtain a subset of C_2. Hence, C_1 makes C_2 redundant—that is, after deriving C_1, we may safely delete C_2.

A *derivation* from a closure set N_0 is a sequence of closure sets $N_0, N_1, \ldots, N_i$, where $N_i = N_{i-1} \cup \{C\}$ and C is derived by applying a $\mathcal{BS}$ inference rule to premises from N_{i-1}, or $N_i = N_{i-1} \setminus \{C\}$ and C is redundant in N_{i-1}. $\mathcal{BS}$ is a sound and complete refutation procedure: if no nonredundant inference is applicable to a closure set N_i (that is, if N_i is *saturated*), then N_0 is unsatisfiable if and only if N_i contains the empty closure.

In our proofs, we use an additional *splitting* inference rule: if a closure $C \cdot \sigma$ consists of n parts $C_i \cdot \sigma$, $2 \leq i \leq n$, such that $C_i\sigma$ and $C_j\sigma$ do not share a common variable for $i \neq j$, then one can separately assume that some $C_i \cdot \sigma$ is true. A *derivation with splitting* from a set of closures N_0 is a finitely branching tree whose nodes are closure sets and whose root is N_0. The children of a node N_i are closure sets obtained by applying an inference rule or a redundancy elimination rule to N_i. A set of closures N_0 is satisfiable if and only if each derivation from N_0 contains a saturated node N_i not containing the empty closure.

2.3. Disjunctive Datalog

A *disjunctive datalog program with equality* P is a finite set of *rules* of the form $A_1 \vee ... \vee A_n \leftarrow B_1, ..., B_m$, where A_i and B_j are *datalog atoms* $A(t_1, \ldots, t_n)$ or $t_1 \approx t_2$, and t_i are variables or constants. In a rule, A_i are the *head* and B_i are the *body atoms*. Each rule must be *safe*;

Table II. Inference Rules of Basic Superposition

Positive superposition:
$$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \approx v) \cdot \rho}{(C \vee D \vee w[t]_p \approx v) \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, w\rho|_p)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$, (iii) $(s \approx t)\cdot\theta$ is strictly eligible for superposition in $(C \vee s \approx t) \cdot \theta$, (iv) $(w \approx v) \cdot \theta$ is strictly eligible for superposition in $(D \vee w \approx v) \cdot \theta$, (v) $s\theta \approx t\theta \not\succeq w\theta \approx v\theta$, (vi) $w|_p$ is not a variable.

Negative superposition:
$$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \not\approx v) \cdot \rho}{(C \vee D \vee w[t]_p \not\approx v) \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, w\rho|_p)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$, (iii) $(s \approx t)\cdot\theta$ is strictly eligible for superposition in $(C \vee s \approx t) \cdot \theta$, (iv) $(w \not\approx v) \cdot \theta$ is eligible for resolution in $(D \vee w \not\approx v) \cdot \theta$, (v) $w|_p$ is not a variable.

Reflexivity resolution:
$$\frac{(C \vee s \not\approx t) \cdot \rho}{C \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, t\rho)$ and $\theta = \rho\sigma$, (ii) $(s \not\approx t) \cdot \theta$ is eligible for resolution in $(C \vee s \not\approx t) \cdot \theta$.

Equality factoring:
$$\frac{(C \vee s \approx t \vee s' \approx t') \cdot \rho}{(C \vee t \not\approx t' \vee s' \approx t') \cdot \theta}$$

where (i) $\sigma = \mathsf{MGU}(s\rho, s'\rho)$ and $\theta = \rho\sigma$, (ii) $t\theta \not\succeq s\theta$ and $t'\theta \not\succeq s'\theta$, (iii) $(s \approx t) \cdot \theta$ is eligible for superposition in $(C \vee s \approx t \vee s' \approx t') \cdot \theta$.

Ordered Hyperresolution:
$$\frac{E_1 \ldots E_n \quad N}{(C_1 \vee \ldots \vee C_n \vee D) \cdot \theta}$$

where (i) E_i are closures of the form $(C_i \vee A_i) \cdot \rho$, for $1 \leq i \leq n$, (ii) N is a closure of the form $(D \vee \neg B_1 \vee \ldots \vee \neg B_n) \cdot \rho$, (iii) σ is the most general substitution such that $A_i\theta = B_i\theta$ for $1 \leq i \leq n$ and $\theta = \rho\sigma$, (iv) each $A_i \cdot \theta$ is strictly eligible for superposition in E_i, (v) either all $\neg B_i \cdot \theta$ are selected, or nothing is selected, $n = 1$, and $\neg B_1 \cdot \theta$ is maximal w.r.t. $D \cdot \theta$.

that is, each variable occurring in the rule must occur in at least one body atom. A *fact* is a rule with $m = 0$. For the semantics, we take a rule to be equivalent to the clause $A_1 \vee ... \vee A_n \vee \neg B_1 \vee ... \vee \neg B_m$. We consider only Herbrand models over all constants from P in which $\approx$ is interpreted as a congruence relation [15]. We say that a model M of P is *minimal* if there is no model M' of P such that $M' \subsetneq M$. A ground literal A is a *cautious answer* of P (written $P \models_c A$) if A is true in all minimal models of P. For positive ground atoms, first-order entailment coincides with cautious entailment.

3. Reducing $\mathcal{SHIQ}$ to Disjunctive Datalog

For a $\mathcal{SHIQ}$ knowledge base KB, our goal is to compute a disjunctive datalog program $\mathsf{DD}(KB)$ such that, for α of the form $A(a)$ or $R(a,b)$, $KB \models \alpha$ if and only if $\mathsf{DD}(KB) \models \alpha$. In other words, KB and $\mathsf{DD}(KB)$ should entail the same set of positive atomic ground facts. We can thus use $\mathsf{DD}(KB)$ instead of KB for query answering, and in doing so we can apply all optimization techniques known in deductive databases.

As pointed out by Borgida, many description logics are fragments of first-order logic [8] (see Definition 2), so the translation may seem to be trivial. Consider, however, the following knowledge base:

$$KB = \{A \sqsubseteq \exists R.A, \exists R.\exists R.A \sqsubseteq B, A(a)\} \tag{1}$$

A naïve solution to our problem is to translate KB into first-order logic as $\pi(KB)$, skolemize it, translate it into conjunctive normal form, and rewrite the resulting set of clauses into rules. This produces the following logic program $\mathsf{LP}(KB)$:

$$R(x, f(x)) \leftarrow A(x) \tag{2}$$

$$A(f(x)) \leftarrow A(x) \tag{3}$$

$$B(x) \leftarrow R(x,y), R(y,z), A(z) \tag{4}$$

$$A(a) \tag{5}$$

Clearly, KB and $\mathsf{LP}(KB)$ entail the same set of ground facts; however, $\mathsf{LP}(KB)$ contains a recursive rule (3) with a function symbol in its head atom. This causes problems when $\mathsf{LP}(KB)$ is used to answer queries. Namely, well-known query evaluation techniques do not necessarily terminate on $\mathsf{LP}(KB)$; for example, the bottom-up saturation derives $A(f(a))$, $R(a, f(a))$, $A(f(f(a)))$, $R(f(a), f(f(a)))$, $B(a)$, and so on. Since we keep deriving ever deeper facts, the algorithm never terminates. Note that we need all previously derived facts to derive the ground fact $B(a)$, and that we do not know a priori when all relevant ground facts have been derived.

This problem could be solved by employing an appropriate cycle detection mechanism. For example, Hustadt and Schmidt use such an approach to derive a hyperresolution decision procedure for the basic description logic $\mathcal{ALC}$ [25]. Such an algorithm for evaluating queries in $\mathsf{LP}(KB)$ would take us away from our original goal of applying deductive database optimization techniques to description logics. The entire procedure could be understood as an alternative syntactic notation for the tableau calculus, so it is still unclear how to combine it with the optimization techniques of deductive databases.

To avoid potential problems with termination, our goal is to derive a true disjunctive datalog program $\mathsf{DD}(KB)$—that is, a program without function symbols. For such a program, queries can be evaluated using any standard technique; furthermore, all existing optimization techniques known in deductive databases can be applied directly. Hence, the main problem that we deal with is the elimination of function symbols from $\mathsf{LP}(KB)$.

3.1. The General Idea

In order to obtain the desired reduction, we start with a slightly simpler task of deriving a program $\mathsf{DD}(KB)$ that is satisfiable if and only if KB is satisfiable. The principle for proving the equisatisfiability of KB and $\mathsf{DD}(KB)$ is relatively straightforward. Let us assume that unsatisfiability of KB can be demonstrated by a refutation in a sound and complete calculus $\mathcal{C}$. We show that $\mathsf{DD}(KB)$ can simulate each inference of $\mathcal{C}$ on KB and, conversely, that a refutation in $\mathsf{DD}(KB)$ can be reduced to a refutation by $\mathcal{C}$ in KB.

In order to derive a sound, complete, and terminating algorithm from the high-level idea outlined in the previous paragraph, we must select an appropriate calculus $\mathcal{C}$, capable of deciding satisfiability of KB. Positive disjunctive datalog is strongly related to clausal first-order logic, so the simulation of the inferences of $\mathcal{C}$ in disjunctive datalog should be easier if $\mathcal{C}$ is a clausal refutation calculus. Therefore, we have chosen $\mathcal{C}$ to be basic superposition and, in Section 3.2, we sketch our decision procedure for $\mathcal{SHIQ}$ by $\mathcal{BS}$ from [22, 23].

Based on this decision procedure, our algorithm computes $\mathsf{DD}(KB)$ by the following steps:

- Due to technical reasons, transitivity axioms are first eliminated using a polynomial transformation. In the remaining sections, we thus consider only $\mathcal{ALCHIQ}$ knowledge bases.
- The knowledge base is translated into an equisatisfiable set of closures $\Xi(KB)$, as defined in Section 3.2. The closures in $\Xi(KB)$ obtained by transforming the TBox and RBox axioms of KB are then saturated using $\mathcal{BS}$; let $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ denote the saturated set of closures.
- As described in Section 3.3, function symbols are eliminated from $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$, producing a function-free set of closures $\mathsf{FF}(KB)$. As shown in Lemma 14, this transformation does not affect satisfiability.

– In order to reduce the size of the datalog program, some irrelevant closures are removed from $\mathsf{FF}(KB)$, as explained in Section 3.4, producing a set of closures $\mathsf{FF}_\mathsf{R}(KB)$. As shown in Lemma 16, this transformation also does not affect satisfiability.

– Finally, $\mathsf{FF}_\mathsf{R}(KB)$ is transformed into a disjunctive datalog program $\mathsf{DD}(KB)$, as described in Section 3.5. This transformation is straightforward: it suffices to transform each closure into the equivalent sequent form. Theorem 18 summarizes the properties of the resulting datalog program.

We apply this algorithm to several examples in Section 3.6, and we discuss certain properties of our approach in Section 3.7.

3.2. Deciding Satisfiability of KB by $\mathcal{BS}$

We now sketch our procedure for deciding satisfiability of a $\mathcal{SHIQ}$ knowledge base KB by basic superposition. We present here just the main results and proof sketches; the full proofs are given in [22, 23, 28].

Eliminating Transitivity Axioms. Transitivity axioms are translated into closures of the form $\neg R(x,y) \vee \neg R(y,z) \vee R(x,z)$. Such closures do not contain so-called *covering* literals (i.e., literals containing all variables of a closure), and are therefore difficult to handle by resolution [26]. Hence, we preprocess KB into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(KB)$. In short, this transformation replaces each transitivity axiom $\mathsf{Trans}(S)$ with axioms of the form $\forall R.C \sqsubseteq \forall S.(\forall S.C)$, for each R with $S \sqsubseteq^* R$ and each concept C occurring in KB. This transformation is polynomial. KB and $\Omega(KB)$ entail the same ground facts concerning simple roles; however, this is not true for complex roles. This is why we allow only simple roles in ABox assertions $\neg S(a,b)$ in Definition 1. In the rest of this paper, we consider only $\mathcal{ALCHIQ}$ knowledge bases; our results apply to $\mathcal{SHIQ}$ knowledge bases as well, provided that only simple roles are allowed in queries. For a precise definition of this transformation and for proof of its correctness, please refer to [28, Section 5.2].

Translation into Closures. To decide satisfiability of $\pi(KB)$, we transform it into a clausal form. A straightforward transformation of $\pi(KB)$ into conjunctive normal form might exponentially increase the formula size and could destroy the structure of the formula. Therefore, before clausification, we apply the *structural transformation* [35, 32, 5] (also known as *renaming*) to KB, which is well-known to be polynomial, and the result of which can be translated polynomially into closures.

We use Cls to denote the standard operator for translating a first-order formula into clausal form by skolemization and rewriting into conjunctive normal form [32].

DEFINITION 3. *Let C be a concept and Λ a function assigning to C the set of positions $p \neq \epsilon$ such that $C|_p$ is not a literal concept and, for all positions q below p, $C|_q$ is a literal concept. The operator $\mathsf{Def}(C)$ is recursively defined as follows. If $\Lambda(C) = \emptyset$, then $\mathsf{Def}(C) = \{C\}$; otherwise, choose $p \in \Lambda(C)$ and let $\mathsf{Def}(C)$ be as follows, for Q a new, globally unique atomic concept:*

$$\mathsf{Def}(C) = \begin{cases} \{\neg Q \sqcup C|_p\} \cup \mathsf{Def}(C[Q]_p) & \text{if } \mathsf{pol}(C,p) = 1 \\ \{Q \sqcup \neg C|_p\} \cup \mathsf{Def}(C[Q]_p) & \text{if } \mathsf{pol}(C,p) = -1 \end{cases}$$

The clausification operator Cls is extended to concepts as follows:

$$\mathsf{Cls}(C) = \bigcup_{D \in \mathsf{Def}(C)} \mathsf{Cls}(\forall x : \pi_x(D))$$

For KB an extensionally reduced $\mathcal{ALCHIQ}$ knowledge base, $\Xi(KB)$ is the smallest set such that

- *if a role name R occurs in KB, then $\mathsf{Cls}(\pi(R)) \subseteq \Xi(KB)$;*
- *$\mathsf{Cls}(\pi(\alpha)) \subseteq \Xi(KB)$ for each RBox or ABox axiom α of KB; and*
- *if $C \sqsubseteq D \in KB_{\mathcal{T}}$, then $\mathsf{Cls}(\neg C \sqcup D) \subseteq \Xi(KB)$.*

If KB is not extensionally reduced, then $\Xi(KB) = \Xi(KB')$, where KB' is the extensionally reduced knowledge base obtained from KB as described in Section 2.1.

We call the closures of types from Table III *$\mathcal{ALCHIQ}$-closures*. The following lemma summarizes the properties of $\Xi(KB)$:

LEMMA 4 ([28, Lemma 5.3.2]). *An $\mathcal{ALCHIQ}$ knowledge base KB is satisfiable if and only if $\Xi(KB)$ is satisfiable. Furthermore, each closure from $\Xi(KB)$ is of one of the types given in Table III.*

Decomposition. As we have shown in [23], if KB contains number restrictions on roles that have subroles, saturating $\Xi(KB)$ by $\mathcal{BS}$ need not terminate. To remedy that, we introduce *decomposition*—an additional inference rule that replaces each conclusion of the form below left with the two closures to the right, where t is an arbitrary term, and $Q_{S,f}$ is a predicate that does not occur in $\Xi(KB)$ and is unique for a pair of role and function symbols S and f; we use $\mathcal{BS}^+$ to denote the superposition

Table III. Types of $\mathcal{ALCHIQ}$-Closures

1	$\neg R(x,y) \vee \mathsf{Inv}(R)(y,x)$
2	$\neg R(x,y) \vee S(x,y)$
3	$\mathbf{P}(x) \vee R(x, \langle f(x) \rangle)$
4	$\mathbf{P}(x) \vee R([f(x)], x)$
5	$\mathbf{P_1}(x) \vee \mathbf{P_2}(\langle \mathbf{f}(x) \rangle) \vee \bigvee \langle f_i(x) \rangle \bowtie \langle f_j(x) \rangle$
6	$\mathbf{P_1}(x) \vee \mathbf{P_2}([g(x)]) \vee \mathbf{P_3}(\langle \mathbf{f}([g(x)]) \rangle) \vee \bigvee \langle t_i \rangle \bowtie \langle t_j \rangle$ where t_i and t_j are either of the form $f([g(x)])$ or of the form x
7	$\mathbf{P_1}(x) \vee \bigvee_{i=1}^{n} \neg R(x, y_i) \vee \bigvee_{i=1}^{n} \mathbf{P_2}(y_i) \vee \bigvee_{i=1}^{n} \overset{n}{\underset{j=i+1}{}} y_i \approx y_j$
8	$\mathbf{R}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle) \vee \mathbf{P}(\langle \mathbf{t} \rangle) \vee \bigvee \langle t_i \rangle \bowtie \langle t_j \rangle$ where t, t_i, and t_j are either a constant b or a term $f_i([a])$

Note: The symbol $\bowtie$ stands for either $\approx$ or $\not\approx$, $\langle t \rangle$ means that the term t may, but need not be marked, $\mathbf{P}(t)$ is a shortcut for a (possibly empty) disjunction of the form $(\neg)P_1(t) \vee \ldots \vee (\neg)P_n(t)$, and $\mathbf{P}(\mathbf{f}(x))$ is a shortcut for a disjunction for the form $\mathbf{P_1}(f_1(x)) \vee \ldots \vee \mathbf{P_m}(f_m(x))$.

calculus extended with the decomposition rule. The completeness proof of $\mathcal{BS}^+$ and examples of its usage can be found in [23].

$$D \cdot \rho \vee R([t], [f(t)]) \rightsquigarrow \begin{array}{rcl} D \cdot \rho & \vee & Q_{R,f}([t]) \\ \neg Q_{R,f}(x) & \vee & R(x, [f(x)]) \end{array}$$

$$D \cdot \rho \vee R([f(t)], [t]) \rightsquigarrow \begin{array}{rcl} D \cdot \rho & \vee & Q_{\mathsf{Inv}(R),f}([t]) \\ \neg Q_{\mathsf{Inv}(R),f}(x) & \vee & R([f(x)], x) \end{array}$$

Parameters for $\mathcal{BS}^+$. The following definition specifies the parameters for $\mathcal{BS}^+$ required in order to obtain a decision procedure for $\mathcal{ALCHIQ}$:

DEFINITION 5. *$\mathcal{BS}^+_{DL}$ is the $\mathcal{BS}^+$ calculus where (*i*) the term ordering $\succ$ is a lexicographic path ordering [4] induced by a total precedence $>$ such that $f > c > P > Q_{R,f} > \mathsf{tt}$, for each function symbol f, constant symbol c, predicate symbol P, and predicate $Q_{R,f}$; and (*ii*) the selection function selects every negative binary literal.*

The parameters used in Definition 5 are compatible with basic superposition and decomposition, so, by [6, 31] and [23], $\mathcal{BS}^+_{DL}$ is a sound and complete calculus. Also, for a finite signature, it is clear that $\mathcal{BS}^+_{DL}$ can introduce at most a quadratic number of predicates $Q_{R,f}$.

Saturating $\Xi(KB)$ *by* $\mathcal{BS}^+_{DL}$. Since $\mathcal{BS}^+_{DL}$ is sound and complete, we can use it to check satisfiability of $\Xi(KB)$. In order to obtain a decision procedure, we must only show that each saturation of $\Xi(KB)$ terminates. We do this in a proof-theoretic way. We start by showing that any nonredundant conclusion of each inference by $\mathcal{BS}^+_{DL}$ on $\mathcal{ALCHIQ}$-closures is an $\mathcal{ALCHIQ}$-closure.

LEMMA 6 ([28, Lemma 5.3.6]). *Let N be a set of $\mathcal{ALCHIQ}$-closures and $C \cdot \rho$ a closure obtained by applying a $\mathcal{BS}^+_{DL}$ inference to premises from N. Then, either $C \cdot \rho$ is redundant in N, or it is an $\mathcal{ALCHIQ}$-closure.*

Proof. [Sketch.] Table IV shows the types of $\mathcal{BS}^+_{DL}$ inferences that can be applied to $\mathcal{ALCHIQ}$-closures. The inferences are shown using the following notation:

(first premise type) first premise			
(other premise type)	other premise(s)	inf. rule	(result type)
...			

Some inferences take several premises: 3^+ denotes one or more premises of type 3; $4 + 3^*$ denotes a premise of type 4 and zero or more premises of type 3; and $3^* + 8^+$ denotes zero or more premises of type 3 and one or more premises of type 8. Furthermore, HR denotes the hyperresolution inference rule, S the (positive or negative) superposition inference rule, and S+D denotes superposition followed by decomposition. The terms and literals participating in the inference are underlined. Inferences for most closure types are symmetric (for example, superposition from a closure of type 5 can be performed into a closure of type 6 and vice versa); for the sake of brevity, we present in the table only one direction. By analyzing the application of all possible inferences by $\mathcal{BS}^+_{DL}$ to all types of $\mathcal{ALCHIQ}$-closures, we can see that all inferences derive an $\mathcal{ALCHIQ}$-closure. □

Next, we show that the number of $\mathcal{ALCHIQ}$-closures is finite for a finite signature of $\Xi(KB)$.

LEMMA 7 ([28, Lemma 5.3.10]). *Let $N_0, N_1, \ldots, N_i$ be a derivation by $\mathcal{BS}^+_{DL}$ from $\Xi(KB)$ and C a closure from some N_i. Then, $|C|$ is at most polynomial in $|KB|$, and $|N_i|$ is at most exponential in $|KB|$, for unary coding of numbers in the input.*

Proof. [Sketch.] Since no inference produces a closure of type 7, the result of any inference is a closure containing at most one variable. Let f be the number of function symbols, i the number of individuals, and

Table IV. Inferences with $\mathcal{ALCHIQ}$-Closures

(1 or 2) $\underline{\neg R(x,y)} \vee \mathsf{Inv}(R)(y,x)$ or $\underline{\neg R(x,y)} \vee S(x,y)$		
(3) $\mathbf{P}(x) \vee \underline{R(x, \langle f(x) \rangle)}$	HR	(3 or 4)
(4) $\mathbf{P}(x) \vee \underline{R([f(x)], x)}$	HR	(4 or 3)
(8) $C \cdot \rho \vee \underline{R(a,b)}$	HR	(8)
(5) $\mathbf{P_1}(x) \vee \underline{\neg A(x)}$		
(5) $C \cdot \rho \vee \underline{A(x)}$ or $C \cdot \rho \vee \underline{A(\langle f(x) \rangle)}$	HR	(5)
(6) $C \cdot \rho \vee \underline{A(\langle f([g(x)]) \rangle)}$	HR	(6)
(8) $C \cdot \rho \vee \underline{A(\langle d \rangle)}$ or $C \cdot \rho \vee \underline{A(\langle f([d]) \rangle)}$	HR	(8)
(5) $\mathbf{P_1}(x) \vee \mathbf{P_2}(\mathbf{f}(x)) \vee \bigvee \langle f_i(x) \rangle \bowtie \langle f_j(x) \rangle \vee \underline{\neg A(f(x))}$		
(6) $C \cdot \rho \vee \underline{A(\langle f([g(x)]) \rangle)}$	HR	(6)
(8) $C \cdot \rho \vee \underline{A(\langle f([a]) \rangle)}$	HR	(8)
(5) $\mathbf{P_1}(x) \vee \mathbf{P_2}(\mathbf{f}(x)) \vee \bigvee \langle f_i(x) \rangle \bowtie \langle f_j(x) \rangle \vee \underline{[f(x)]} \approx [h(x)]$		
(3) $\mathbf{P}(x) \vee R(x, \underline{f(x)})$	S+D	(3+5)
(5) $C \cdot \rho \vee \underline{A(f(x))}$ or $C \cdot \rho \vee \underline{f(x)} \bowtie \langle h'(x) \rangle$	S	(5)
(6) $C \cdot \rho \vee \underline{f([g(x)])} \bowtie \langle t' \rangle$	S	(6)
(8) $C \cdot \rho \vee A(\underline{f([a])})$ or $C \cdot \rho \vee f([a]) \bowtie \langle t' \rangle$	S	(8)
(6) $\mathbf{P_1}(x) \vee \mathbf{P_2}([g(x)]) \vee \mathbf{P_3}(\langle \mathbf{f}([g(x)]) \rangle) \vee \bigvee \langle t_i \rangle \bowtie \langle t_j \rangle \vee \underline{\neg A(\langle f([g(x)]) \rangle)}$		
(6) $C \cdot \rho \vee \underline{A(\langle f([g(x)]) \rangle)}$	HR	(6)
(6) $\mathbf{P_1}(x) \vee \mathbf{P_2}([g(x)]) \vee \mathbf{P_3}(\langle \mathbf{f}([g(x)]) \rangle) \vee \bigvee \langle t_i \rangle \bowtie \langle t_j \rangle \vee \underline{[f(g(x))]} \approx t$		
(3) $\mathbf{P}(x) \vee R(x, \underline{f(x)})$	S+D	(3+6 or 4+6)
(5) $C \cdot \rho \vee A(\underline{f(x)})$ or $C \cdot \rho \vee \underline{f(x)} \bowtie \langle h'(x) \rangle$	S	(6)
(6) $C \cdot \rho \vee A(\underline{f([g(x)])})$ or $C \cdot \rho \vee \underline{f([g(x)])} \bowtie \langle t' \rangle$	S	(6)
(7) $\mathbf{P_1}(x) \vee \bigvee_{i=1}^{n} \underline{\neg R(x, y_i)} \vee \bigvee_{i=1}^{n} \mathbf{P_2}(y_i) \vee \bigvee_{i=1}^{n} \bigvee_{j=i+1}^{n} y_i \approx y_j$		
(3^+) $\mathbf{P^{f_i}}(x) \vee \underline{R(x, \langle f_i(x) \rangle)}$	HR	(5)
$(4 + 3^*)$ $\mathbf{P^g}(x) \vee \underline{R([g(x)], x)}$ and $\mathbf{P^{f_i}}(x) \vee \underline{R(x, \langle f_i(x) \rangle)}$	HR	(6)
$(3^* + 8^+)$ $\mathbf{P^{f_i}}(x) \vee \underline{R(x, \langle f_i(x) \rangle)}$ and $C \cdot \rho \vee \underline{R(a,b)}$	HR	(8)
(8) $\mathbf{R}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle) \vee \mathbf{P}(\langle \mathbf{t} \rangle) \vee \bigvee \langle t_i \rangle \bowtie \langle t_j \rangle \vee \underline{[f(a)]} \approx t$		
(3) $\mathbf{P}(x) \vee R(x, \underline{f(x)})$	S+D	(3+8)
(5) $C \cdot \rho \vee \underline{A(f(x))}$ or $C \cdot \rho \vee \underline{f(x)} \bowtie \langle h'(x) \rangle$	S	(8)
(8) $C \cdot \rho \vee A(\underline{f([a])})$ or $C \cdot \rho \vee f([a]) \bowtie \langle t' \rangle$	S	(8)

Remaining inferences involve closures of type 8 and produce a closure of type 8. Equality factoring is applicable only to literals of the form $f(t) \approx g(t)$ and produces a closure of the same type. Reflexivity resolution can only eliminate literals of the form $f(t) \not\approx f(t)$ and also produces a closure of the same type.

p the number of (unary and binary) predicates. Since the numbers are coded in unary, f, i, and p are polynomial in $|KB|$ (remember that the number of predicates $Q_{R,f}$ is at most quadratic in $|KB|$). Each literal contains at most four function symbols, either one variable or at most two individuals, at most two marked positions, and is either positive or negative. Hence, the number of different literals is at most $2 \cdot 4p(i+1)^2(f+1)^4$, which is polynomial in $|KB|$. Each nonredundant closure contains a subset of these literals, so there are exponentially many different closures. □

One can show that each saturation derives at most exponentially many redundant closures. Together with lemmas 6 and 7, we have that, in the worst case, we can derive all possible $\mathcal{ALCHIQ}$-closures in a saturation, after which the saturation terminates:

THEOREM 8 ([28, Theorems 5.3.11 and 5.4.8]). *For an $\mathcal{ALCHIQ}$ knowledge base KB, saturation of $\Xi(KB)$ by $\mathcal{BS}^+_{DL}$ with eager elimination of redundancy decides satisfiability of KB, and runs in time exponential in $|KB|$, for unary coding of numbers in input.*

The following corollary follows from the proof of Lemma 6. Namely, only maximal literals from a closure can participate in an inference, so a term of the form $f(g(x))$ from a closure of type 6 does not unify with a term from a closure of type 8 (which always has the form a or $f(a)$); a similar argument holds for closures of type 4.

COROLLARY 9 ([28, Corollary 5.3.8]). *If a closure of type 8 participates in a $\mathcal{BS}^+_{DL}$ inference with other $\mathcal{ALCHIQ}$-closures, then the unifier σ contains only ground mappings of the form $x \mapsto a$ and $x \mapsto f(b)$, and the conclusion is a closure of type 8. Furthermore, a closure of type 8 cannot participate in an inference with a closure of type 4 or 6.*

3.3. Eliminating Function Symbols

We now show how to eliminate function symbols from $\Xi(KB)$.

DEFINITION 10. *For KB an extensionally reduced $\mathcal{ALCHIQ}$ knowledge base, let $\mathsf{gen}(KB)$ contain a closure $\neg Q_{R,f}(x) \vee R(x, [f(x)])$ for each role R and function symbol f occurring in $\Xi(KB)$.[2] Furthermore, let $\Gamma_{\mathcal{TR}g} = \Xi(KB_{\mathcal{T}} \cup KB_{\mathcal{R}}) \cup \mathsf{gen}(KB)$. With $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g})$ we denote*

[2] As discussed in Section 3.2, $Q_{R,f}$ is a predicate introduced by decomposition and is unique for all pairs of R and f.

the relevant set of saturated closures—*that is, closures of types 1, 2, 3, 5, and 7 obtained by saturating* $\Gamma_{\mathcal{TR}g}$ *by* $\mathcal{BS}^+_{DL}$ *with eager application of redundancy elimination rules. Finally, let* $\Gamma = \mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_{\mathcal{A}})$.

We now show that eliminating closures of types 4 and 6 does not affect satisfiability.

LEMMA 11. *KB is unsatisfiable if and only if* $\Gamma(KB)$ *is unsatisfiable, for KB an extensionally reduced* $\mathcal{ALCHIQ}$ *knowledge base.*

Proof. Let $\Gamma' = \Xi(KB) \cup \mathsf{gen}(KB)$. Each closure $C \in \mathsf{gen}(KB)$ contains a new predicate $Q_{R,f}$ unique for C, so any interpretation of $\Xi(KB)$ can be extended to an interpretation of Γ'. Hence, KB is equisatisfiable with Γ' by Lemma 4. Since $\mathcal{BS}^+_{DL}$ is sound and complete, Γ' is unsatisfiable if and only if there is a derivation by $\mathcal{BS}^+_{DL}$ of the empty closure from Γ'. The order in which inferences are performed in a derivation $N_0, N_1, \ldots$ from Γ' can be chosen don't-care nondeterministically, so we perform all nonredundant inferences among closures from $\Gamma_{\mathcal{TR}g}$ first. Let $N_m = \mathsf{Sat}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_{\mathcal{A}})$ be the intermediate set of closures obtained in such a derivation, where $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ is the saturated set of closures.

Each closure set N_i with $i > m$ is obtained from N_{i-1} by an inference involving at least one closure not in N_m. All nonredundant inferences between closures of type other than 8 have been performed in N_m, and, by Corollary 9, each inference involving a closure of type 8 produces a closure of type 8. Furthermore, if a conclusion of an inference is decomposed into closures of types 3 and 8, the closure of type 3 is contained in $\mathsf{gen}(KB)$, so only the closure of type 8 is added to N_i. Hence, $N_i \setminus N_m$ contains only closures of type 8.

Furthermore, by Corollary 9, a closure of type 4 and 6 can never participate in an inference with a closure of type 8, and it cannot be used to derive a closure in $N_i \setminus N_m$ for $i > m$. Hence, N_m can safely be replaced with Γ in the derivation: all inferences by $\mathcal{BS}^+_{DL}$ using closures from N_m can be performed using closures from Γ as well. □

If KB does not use number restrictions, further optimizations are possible. Then, closures of types 3 or 5 containing a function symbol cannot participate in an inference with a closure of type 8. Hence, closures of types 3 and 5 can also be eliminated from the saturated set; that is, they need not be included in $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TR}g})$.

We now show how to eliminate function symbols from closures in Γ:

DEFINITION 12. *The operator λ maps terms to terms as follows:* (i) $\lambda(a) = a$; (ii) $\lambda(f(a)) = a_f$, *for a_f a new unique constant;*[3] (iii) $\lambda(x) = x$; (iv) $\lambda(f(x)) = x_f$, *for x_f a new unique variable.*

We extend λ to $\mathcal{ALCHIQ}$-closures such that, for a closure C, $\lambda(C)$ is the function-free closure computed as follows:

1. *Each term t in the closure is replaced with $\lambda(t)$.*
2. *For each variable x_f introduced in step 1, the literal $\neg S_f(x, x_f)$ is added to the closure, where S_f is a new predicate unique for f.*
3. *If, after steps 1 and 2, a variable x occurs in a positive, but not in a negative literal, the literal $\neg HU(x)$ is added to the closure.*

For a substitution σ, let $\lambda(\sigma)$ denote the substitution obtained from σ by replacing each assignment $x \mapsto t$ with $x \mapsto \lambda(t)$. With λ^- we denote the inverse of λ (that is, $\lambda^-(\lambda(\alpha)) = \alpha$ for any term, closure, or any substitution α).[4]

For an extensionally reduced $\mathcal{ALCHIQ}$ knowledge base KB, the function-free version *of $\Xi(KB)$, written $\mathsf{FF}(KB)$, is defined as follows:*

$$\begin{aligned}
\mathsf{FF}_\lambda(KB) &= \{\lambda(C) \mid C \in \mathsf{Sat}_\mathsf{R}(\Gamma_{\mathcal{TR}g})\} \\
\mathsf{FF}_{Succ}(KB) &= \{S_f(a, a_f) \mid \textit{for each } a \textit{ and } f \textit{ from } \Xi(KB)\} \\
\mathsf{FF}_{HU}(KB) &= \{HU(a), HU(a_f) \mid \textit{for each } a \textit{ and } f \textit{ from } \Xi(KB)\} \\
\mathsf{FF}(KB) &= \mathsf{FF}_\lambda(KB) \cup \mathsf{FF}_{Succ}(KB) \cup \mathsf{FF}_{HU}(KB) \cup \Xi(KB_\mathcal{A})
\end{aligned}$$

LEMMA 13. *Nonground negative equality literals occur in $\mathsf{FF}(KB)$ only in disjunctions of the form $x_f \not\approx x_g \vee \neg S_f(x, x_f) \vee \neg S_g(x, x_g)$.*

Proof. The closure $\lambda(C)$ can contain nonground equalities only for C of type 5. Negative equality literals occur in such closures only as $f(x) \not\approx g(x)$, so $\lambda(C)$ contains $x_f \not\approx x_g \vee \neg S_f(x, x_f) \vee \neg S_g(x, x_g)$. □

We now show that $\mathsf{FF}(KB)$ can simulate all inferences by $\mathcal{BS}^+_{DL}$ on Γ (c.f. Table IV) and vice versa, which gives us the following key lemma:

LEMMA 14. *KB is unsatisfiable if and only if $\mathsf{FF}(KB)$ is unsatisfiable.*

Proof. Due to Lemma 11, we just need to show that the closure sets $\Gamma = \mathsf{Sat}_\mathsf{R}(\Gamma_{\mathcal{TR}g}) \cup \Xi(KB_\mathcal{A})$ and $\mathsf{FF}(KB)$ are equisatisfiable.

($\Leftarrow$) Let $\mathcal{C}$ be the calculus of hyperresolution with superposition and eager splitting, where all negative nonequality literals are selected. Assume that $\mathsf{FF}(KB)$ is unsatisfiable. Then, a derivation B by $\mathcal{C}$ exists, such that the root node is $\mathsf{FF}(KB)$ and the empty closure is derived on

[3] Unique means that, for each f and a, the constant a_f is uniquely defined.

[4] Note that λ is injective, but not surjective, so to make the definition of λ^- correct, we assume that λ^- is applicable only to the range of λ.

all branches. By induction on the depth of B, we construct a derivation B' from Γ by sound inference steps and splitting such that each node N_n in B corresponds one-to-one to a node N'_n in B' satisfying the following property (*): if C is a closure in N_n not of the form $S_f(u, v)$ or $HU(u)$, then N'_n contains the *counterpart closure* $\lambda^-(C)$. To simplify the presentation, we allow the children of a node in B' to be obtained by applying zero or more sound inferences to the parent node. The induction base $n = 0$ is obvious. Now assume that, for some node N_n in B, we have a node N'_n in B' satisfying (*), and consider all possible inferences applicable to closures in N_n.

Superposition can be performed only from a ground closure. Namely, a nonground closure is safe, so it contains negative literals, which are selected and do not contain constants. Furthermore, because of splitting, all ground closures are unit closures, so superposition can be performed only from a closure of the form $s \approx t$. Consider now superposition into a ground unit closure L. If $L = HU(s)$, superposition is redundant: HU is instantiated for each constant occurring in $\mathsf{FF}(KB)$, so the conclusion is already contained in N_n. If $L = S_f(s, u)$ or $L = S_f(u, s)$, the proposition trivially holds. Otherwise, by the induction hypothesis, counterpart closures $\lambda^-(s \approx t)$ and $\lambda^-(L)$ are contained in N'_n, so superposition can be performed on them to derive the required counterpart closure.

Reflexivity resolution can be applied to a ground closure $u \not\approx u$. By the induction hypothesis, the set N'_n contains $\lambda^-(u \not\approx u)$, so reflexivity resolution can be applied to it to derive the required counterpart closure. Reflexivity resolution is not applicable to nonground closures: by Lemma 13, negative literals $x_f \not\approx x_g$ always occur in disjunctions $x_f \not\approx x_g \vee \neg S_f(x, x_f) \vee \neg S_g(x, x_g)$, in which $\neg S_f(x, x_f)$ and $\neg S_g(x, x_g)$ are selected.

Equality factoring is not applicable to a closure from N_n since all positive closures from N_n are ground unit closures.

Consider a hyperresolution with a main premise C, side premises $E_1, \ldots, E_k$, and a unifier σ, resulting in a hyperresolvent H. The side premises E_i are not allowed to contain selected literals, so they are ground unit closures. Furthermore, the closure C is safe, and, by Lemma 13, for each literal of the form $x_f \not\approx x_g$, C contains literals $\neg S_f(x, x_f)$ and $\neg S_g(x, x_g)$, respectively, which are selected. Hence, all variables in C are bound, so H is a ground closure. Let σ' be the substitution such that $x\sigma' = \lambda^-(x\sigma)$ for each variable x not of the form x_f (since the closures from Γ do not contain variables of form x_f, the definition of σ' on them is not relevant). We then perform an instantiation step $C' = \lambda^-(C)\sigma'$ on closures from N'_n. The closures $\lambda^-(C\sigma)$ and C' can differ only at a position p of a variable x_f in C: the closure $C\sigma$

can contain at position p a term v obtained by resolving C with a side premise of the form $S_f(u, v)$, while C' contains at the position p' corresponding to p the term $f(u)$ such that $\lambda^-(v) \neq f(u)$. Let p'_u denote the position of the inner u in $f(u)$. We show that all such discrepancies can be eliminated with sound inferences in N'_n. The literal $S_f(u, v)$ in N_n is obtained from some $S_f(a, a_f)$ by several superposition inference steps. Let us denote with Δ_1 (Δ_2) the sequence of ground unit equalities applied to the first (second) argument of $S_f(a, a_f)$. By the induction hypothesis, for each equality $s_i \approx t_i$ from Δ_1 and Δ_2, the set N'_n contains the corresponding equality $\lambda^-(s_i \approx t_i)$; we denote the sequences of corresponding equalities with Δ'_1 and Δ'_2. We now perform superposition with equalities from Δ'_1 into C' at p'_u in the reverse order. After this, p'_u contains the constant a, and p' contains the term $f(a)$. Hence, we can apply superposition with equalities from Δ'_2 at p' in the original order. After this is done, each position p' contains the term $\lambda^-(v)$. Let C'' denote the result of removing discrepancies at all positions; then, $C'' = \lambda^-(C\sigma)$. For each side premise E_i not of the form $S_f(u, v)$ or $HU(u)$, the set N'_n contains $\lambda^-(E_i)$ by the induction hypothesis, so we can hyperresolve $\lambda^-(E_i)$ premises with C'' to obtain $H' = \lambda^-(H)$. Hence, the counterpart closure is derivable from N'_n.

All ground closures from N_n are unit closures, and no nonground closure in $\mathsf{FF}(KB)$ contains a positive literal with S_f or HU predicates, so a ground nonunit closure C from N_n cannot contain literals of the form $S_f(u, v)$ and $HU(a)$. Hence, if C is of length k and it causes N_n to be split into k child nodes, then $\lambda^-(C)$ is of length k and it causes N'_n to be split into k child nodes, each of them satisfying (*).

Because the nodes of B and B' correspond one-to-one, if B derives the empty closure on all paths from $\mathsf{FF}(KB)$, then B' derives the empty closure on all paths from Γ as well.

($\Rightarrow$) Assume that Γ is unsatisfiable. Then, a derivation B' by $\mathcal{BS}^+_{DL}$ exists, such that Γ is the root closure set and the empty closure is derived. By induction on the length of B', we construct a derivation B from $\mathsf{FF}(KB)$ by sound inference steps such that each closure set N_n in B corresponds one-to-one to a closure set N'_n in B' satisfying the following property (**): if C' is a closure in N'_n, then N_n contains the counterpart closure $C = \lambda(C')$. To simplify the presentation, we allow each N_n, $n > 0$, to be obtained from N_{n-1} by zero or more sound inference steps. The induction base $n = 0$ is trivial. Now assume that, for some closure set N'_n in B', there is a closure set N_n in B satisfying (**), and consider all possible inferences deriving a closure C' from premises $P'_i \in N'_n, 1 \leq i \leq k$.

By the induction hypothesis, N_n contains the counterpart closure P_i of each P'_i. Let σ' be the unifier of the inference deriving C'. Since

all nonredundant inferences among nonground closures were performed while computing $\mathsf{Sat}_{\mathsf{R}}(\Gamma_{\mathcal{TRS}g})$, the inference deriving C' must involve at least one ground closure. By Corollary 9, σ' is ground and it contains only assignments of the form $x_i \mapsto a$ or $x_i \mapsto f(a)$. For $\sigma = \lambda(\sigma')$, we instantiate each P_i into $P_i\sigma$. Consider now possible differences between $P_i\sigma$ and $\lambda(P'_i\sigma')$, apart from the literals involving S_f and HU. A difference can occur if P'_i contains at position p' a term $f(x)$, which is instantiated by σ' to a; then, P_i contains at the position p corresponding to p' the variable x_f which is not instantiated by σ. Hence, at p, the closure $P_i\sigma$ contains x_f and the closure $\lambda(P'_i\sigma')$ contains a_f. By definition of λ, however, the closure P_i then contains a literal $\neg S_f(x, x_f)$, so $P_i\sigma$ contains $\neg S_f(a, x_f)$. This literal can be resolved with $S_f(a, a_f)$ to produce a_f at p. All such differences can be removed iteratively, and, since $\mathsf{FF}(KB)$ contains $HU(a)$ for each constant a, the remaining ground literals involving HU can be resolved away. Hence, $\lambda(P'_i\sigma')$ is derivable from premises in N_n.

In all terms of the form $f(a)$ occurring on B', the inner term a is marked. Hence, superposition inferences are possible only on the outer position of such terms, which correspond via λ to a_f. Therefore, regardless of the inference type, $C = \lambda(C')$ can be derived from $\lambda(P'_i\sigma')$ by the same inference on the corresponding literals.

The result of a superposition inference in B' may be a closure C' containing a literal $R([a], [f(a)])$, which is decomposed into a closure C'_1 of type 8 and a closure C'_2 of type 3. Since $\mathsf{gen}(KB) \subseteq \Gamma$, we have $C'_2 \in \Gamma$, so the conclusion C' should only be replaced with the conclusion C'_1. The decomposition inference rule can be applied to B as well to produce the counterpart closure $C_1 = \lambda(C'_1)$.

As a consequence, if the empty closure is derivable on B', then it is derivable on B as well. □

To illustrate Lemma 14, consider the following example. Let Γ be the following set of closures:

$$D(x) \vee \underline{R(x, f(x))} \tag{6}$$

$$D(x) \vee C(\underline{f(x)}) \tag{7}$$

$$\underline{\neg R(x, y_1)} \vee \underline{\neg R(x, y_2)} \vee y_1 \approx y_2 \tag{8}$$

$$\underline{R(a, b)} \tag{9}$$

Saturation of Γ by $\mathcal{BS}^+_{DL}$ produces the following closures ($R(xx; yy)$ denotes a resolution of xx and yy, $S(xx; yy)$ denotes a superposition of xx into yy, and we omit markers for the sake of readability):

$$D(a) \vee \underline{f(a)} \approx b \quad \text{R(6;8;9)} \tag{10}$$

$$D(a) \vee C(b) \quad \text{S(10;7)} \tag{11}$$

By Definition 12, $\mathsf{FF}(KB)$ contains the following closures:

$$\underline{\neg S_f(x, x_f)} \vee D(x) \vee R(x, x_f) \tag{12}$$

$$\underline{\neg S_f(x, x_f)} \vee D(x) \vee C(x_f) \tag{13}$$

$$\underline{\neg R(x, y_1)} \vee \underline{\neg R(x, y_2)} \vee y_1 \approx y_2 \tag{14}$$

$$\underline{R(a, b)} \tag{15}$$

$$\underline{S_f(a, a_f)} \tag{16}$$

$$\underline{S_f(b, b_f)} \tag{17}$$

Since all inferences among nonground closures in Γ have been performed during saturation, each nonredundant inference by $\mathcal{BS}^+_{DL}$ on Γ must involve at least one ground closure. Our reduction is based on the observation that, for each ground closure C derivable by $\mathcal{BS}^+_{DL}$ on Γ, we can derive a counterpart ground closure $\lambda(C)$ in $\mathsf{FF}(KB)$ as follows:

$$D(a) \vee \underline{R(a, a_f)} \quad \text{R(12;16)} \tag{18}$$

$$D(a) \vee C(\underline{a_f}) \quad \text{R(13;16)} \tag{19}$$

$$D(a) \vee \underline{a_f} \approx b \quad \text{R(14;15;18)} \rightarrow \text{counterpart of (10)} \tag{20}$$

$$D(a) \vee C(b) \quad \text{S(20;19)} \rightarrow \text{counterpart of (11)} \tag{21}$$

Thus, if we can derive the empty closure by saturating Γ by $\mathcal{BS}^+_{DL}$, then we can derive it by saturating $\mathsf{FF}(KB)$ as well. This example also demonstrates that the role of the constants a_f introduced by Definition 12 is merely to simulate in a derivation from $\mathsf{FF}(KB)$ the ground functional terms from a derivation from Γ; there is no deeper semantic relationship between them and the unnamed individuals in a model.

$\mathsf{FF}(KB)$ can be computed once and then used to answer any number of atomic queries. Let α be of form $(\neg)R(a, b)$ or $(\neg)A(a)$ with A an atomic concept, R a role, and a and b occuring in KB. It is then easy to see that $\mathsf{FF}(KB \cup \{\neg\alpha\}) = \mathsf{FF}(KB) \cup \{\neg\alpha\}$. (Note that the constants a and b occurring in the query are assumed to be contained in KB; therefore, $\mathsf{FF}(KB)$ contains the necessary facts about a_f and b_f for each function symbol f.) Hence, Lemma 14 also implies that $KB \models \alpha$ if and only if $\mathsf{FF}(KB) \models \alpha$.

3.4. Removing Irrelevant Closures

The saturation of $\Gamma_{\mathcal{TR}g}$ introduces many closures that are entailed by other closures. For example, for $KB = \{A \sqsubseteq C, C \sqsubseteq B\}$, if literal ordering is such that $C(x) \succ B(x) \succ A(x)$, the saturation derives $\neg A(x) \vee B(x)$; however, this closure is entailed by the premises and is thus not needed for query answering. Hence, we present an optimization

by which we eliminate all such closures and thus reduce the number of rules in the disjunctive datalog program. Intuitively, eliminating such rules is beneficial since it prevents the disjunctive datalog reasoner from drawing the same consequence along different inference paths. Also, this optimization has proven itself to be very important for reducing the size of the translated programs. Our practical experiments have shown that, although the number of closures in $\mathsf{FF}(KB)$ can be quite large, the number of closures left after this optimization is typically just twice the number of TBox axioms [29].

DEFINITION 15. *For $N \subseteq \mathsf{FF}(KB)$, let $C \in N$ be a closure such that $\lambda^-(C)$ is derived in the saturation of $\Gamma_{\mathcal{TR}g}$ by an inference with a unifier σ from premises P_i, $1 \leq i \leq k$. Then, C is* irrelevant w.r.t. *N if the following conditions hold: (*i*) $\lambda^-(C)$ is not derived by the decomposition rule; (*ii*) for each premise P_i, $\lambda(P_i) \in N$; and (*iii*) each variable occurring in any of the $\lambda(P_i\sigma)$ also occurs in C.*

Relevant *is the opposite of irrelevant. Let $C_1, C_2, \ldots, C_n$ be a sequence of closures from $\mathsf{FF}(KB)$ such that $\lambda^-(C_n), \ldots, \lambda^-(C_2), \lambda^-(C_1)$ corresponds to the order in which the closures are derived in saturation of $\Gamma_{\mathcal{TR}g}$. Let $N_0, N_1, \ldots, N_n$ be a sequence of closure sets such that $N_0 = \mathsf{FF}(KB)$, $N_i = N_{i-1}$ if C_i is relevant w.r.t. N_{i-1}, and $N_i = N_{i-1} \setminus \{C_i\}$ if C_i is irrelevant w.r.t. N_{i-1}, for $1 \leq i \leq n$. Then, $\mathsf{FF}_\mathsf{R}(KB) = N_n$ is called the* relevant subset *of $\mathsf{FF}(KB)$.*

LEMMA 16. *$\mathsf{FF}_\mathsf{R}(KB)$ is unsatisfiable if and only if $\mathsf{FF}(KB)$ is unsatisfiable.*

Proof. Let N be a subset of $\mathsf{FF}(KB)$, $C \in N$ a closure that is irrelevant w.r.t. N, and ξ an inference in the saturation of $\Gamma_{\mathcal{TR}g}$ deriving $\lambda^-(C)$ from premises P_i with a unifier σ. Finally, assume $\lambda(P_i) \in N$, $1 \leq i \leq k$. We now show the following property (*): N is satisfiable if and only if $N \setminus \{C\}$ is satisfiable. The ($\Rightarrow$) direction is trivial, since $N \setminus \{C\} \subset N$. For the ($\Leftarrow$) direction, let C' be the closure obtained by an inference from $\lambda(P_i)$ corresponding to ξ. Obviously, each literal not of the form $\neg S_f(x, x_f)$ from C is contained in C' as well. Consider now each literal $\neg S_f(x, x_f)$ in C', stemming from a premise $\lambda(P_i\sigma)$ containing x_f. Since C is irrelevant, it contains x_f, so, by definition of λ, C contains $\neg S_f(x, x_f)$ as well. Thus, $C = C'$, so $\lambda(P_1\sigma), \ldots, \lambda(P_k\sigma) \models C$, and (*) holds.

In the sequence of sets $N_0, N_1, \ldots, N_n$ with $N_0 = \mathsf{FF}(KB)$ and $\mathsf{FF}_\mathsf{R}(KB) = N_n$ from Definition 15, the preconditions of property (*) are satisfied for each $N_i = N_{i-1} \setminus \{C_i\}$, $i \geq 1$ so, by (*), N_i is satisfiable if and only if N_{i-1} is satisfiable. The claim of the lemma now follows by a straightforward induction on i. □

3.5. Reduction to Disjunctive Datalog

Reduction of KB to a disjunctive datalog program is now easy:

DEFINITION 17. *For an extensionally reduced $\mathcal{ALCHIQ}$ knowledge base KB, $\mathsf{DD}(KB)$ is the disjunctive datalog program that contains the rule $A_1\rho \vee \ldots \vee A_n\rho \leftarrow B_1\rho, \ldots, B_m\rho$ for each closure of the form $(A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_m) \cdot \rho$ from $\mathsf{FF}_\mathsf{R}(KB)$. If KB is not extensionally reduced, then $\mathsf{DD}(KB) = \mathsf{DD}(KB')$, where KB' is the extensionally reduced knowledge base obtained from KB as described in Section 2.1.*

THEOREM 18. *For each $\mathcal{ALCHIQ}$ knowledge base KB, the following claims hold:*

1. *KB is unsatisfiable if and only if $\mathsf{DD}(KB)$ is unsatisfiable;*
2. *$KB \models \alpha$ if and only if $\mathsf{DD}(KB) \models_c \alpha$, where α is of the form $A(a)$ or $R(a, b)$, A is an atomic concept, R a role, and a and b from KB;*
3. *$KB \models C(a)$ for a nonatomic concept C and an individual a occurring in KB if and only if $\mathsf{DD}(KB \cup \{C \sqsubseteq Q\}) \models_c Q(a)$ for Q a new atomic concept;*
4. *The number of literals in each rule in $\mathsf{DD}(KB)$ is at most polynomial, the number of rules in $\mathsf{DD}(KB)$ is at most exponential, and $\mathsf{DD}(KB)$ can be computed in time exponential in $|KB|$ for unary coding of numbers in the input.*

Proof. The first claim follows directly from Lemma 16. The second claim follows from the first one: $\mathsf{DD}(KB \cup \{\neg\alpha\}) = \mathsf{DD}(KB) \cup \{\neg\alpha\}$ (since a and b occur in KB, all facts about constants a_f and b_f for each function symbol f are contained in $\mathsf{DD}(KB)$), and $\mathsf{DD}(KB) \cup \{\neg\alpha\}$ is unsatisfiable if and only if $\mathsf{DD}(KB) \models_c \alpha$. Furthermore, $KB \models C(a)$ if and only if $KB \cup \{\neg C(a)\}$ is unsatisfiable, which is the case if and only if $KB \cup \{\neg Q(a), C \sqsubseteq Q\}$ is unsatisfiable. Since Q is atomic and a occurs in KB, the third claim follows from the second one.

By Lemma 7, for each closure $C \in \mathsf{Sat}(\Gamma_{\mathcal{TR}g})$, the number of literals in C is at most polynomial in $|KB|$, and $|\mathsf{Sat}(\Gamma_{\mathcal{TR}g})|$ is at most exponential in $|KB|$. An application of λ to C can be performed in polynomial time. The number of constants a_f added to $\mathsf{DD}(KB)$ is $c \cdot f$, where c is the number of constants, and f the number of function symbols in the signature of $\Xi(KB)$. If numbers are coded in unary, both c and f are polynomial in $|KB|$, so the number of constants a_f is also polynomial in $|KB|$. By Theorem 8, $\mathsf{Sat}(\Gamma_{\mathcal{TR}g})$ can be computed in time exponential in $|KB|$. □

3.6. Examples

We now present several examples that point out important properties of the reduction algorithm. The first three examples do not use number restrictions so, according to the discussion after Lemma 11, we can delete all closures containing function symbols after the saturation of $\Xi(KB)$ by $\mathcal{BS}^+_{DL}$.

Readers familiar with more common approaches to DL reasoning might ask themselves how are the role successors represented in a datalog program. As we show next, datalog programs do not represent them at all. Let $KB_1 = \{C \sqsubseteq \exists R.D\}$; through clausification, we obtain $\Xi(KB_1) = \{\neg C(x) \vee R(x, f(x)), \neg C(x) \vee D(f(x))\}$. The set $\Xi(KB_1)$ is already saturated by $\mathcal{BS}^+_{DL}$. After removing closures with function symbols, we obtain $\mathsf{DD}(KB_1) = \emptyset$, which may be quite confusing: KB_1 implies the existence of at least one R-successor for each member of C, whereas $\mathsf{DD}(KB_1)$ does not reflect that. Theorem 18 is, however, not invalidated. Namely, the individuals introduced by the existential quantifier in KB_1 are unnamed, so they cannot be referred to in the atomic queries or their answers. For an arbitrary extensionally reduced ABox $KB_{\mathcal{A}}$, the knowledge base $KB_1 \cup KB_{\mathcal{A}}$ does not imply any new facts of the form $A(a)$ or $R(a, b)$. Also, note that the models of $\mathsf{DD}(KB_1)$ appear to be unrelated to the models of KB_1 in general; they only coincide on ground facts. This is so because the reduction algorithm is based on a proof-theoretic correspondence between refutations in $\Xi(KB)$ and $\mathsf{DD}(KB)$. The models of KB and $\mathsf{DD}(KB)$ coincide only on positive ground facts, whereas, for unnamed individuals, they do not seem to be related.

In order to draw ground consequences from KB_1, we need more axioms. Let $KB_2 = KB_1 \cup \{D \sqsubseteq \bot\}$, so $\Xi(KB_2) = \Xi(KB_1) \cup \{\neg D(x)\}$. The saturation of $\Xi(KB_2)$ produces a closure $\neg C(x)$, so, eventually, we obtain $\mathsf{DD}(KB_2) = \{\leftarrow C(x), \leftarrow D(x)\}$. This example shows that the reduction is not modular—that is, $\mathsf{DD}(KB_a) \cup \mathsf{DD}(KB_b)$ is not necessarily equal to $\mathsf{DD}(KB_a \cup KB_b)$ for arbitrary knowledge bases KB_a and KB_b.

The key step in our algorithm is the saturation of $\Xi(KB_{\mathcal{T}})$ by $\mathcal{BS}^+_{DL}$. It computes nonground consequences of $\Xi(KB_{\mathcal{T}})$, which ensure that subsequent removal of closures with function symbols does not change the set of ground consequences. These closures are like "macros," because they derive ground facts about objects without expanding their successors. For example, for $KB_3 = \{A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\}$, the set $\Xi(KB_3)$ consists of the following closures:

$$\neg A(x) \vee \underline{R(x, f(x))} \tag{22}$$

$$\neg A(x) \vee \underline{B(f(x))} \tag{23}$$
$$\neg B(x) \vee \underline{C(x)} \tag{24}$$
$$D(x) \vee \neg\underline{R(x,y)} \vee \neg C(y) \tag{25}$$

If literals are ordered as $D(x) \succ C(x) \succ B(x) \succ A(x)$, the saturation of $\Xi(KB_3)$ produces the following additional closures ($R(xx;yy)$ means that a closure is derived by resolving closures xx and yy):

$$\neg A(x) \vee D(x) \vee \underline{\neg C(f(x))} \quad \text{R(22;25)} \tag{26}$$
$$\neg A(x) \vee D(x) \vee \underline{\neg B(f(x))} \quad \text{R(26;24)} \tag{27}$$
$$\neg A(x) \vee D(x) \quad \text{R(27;23)} \tag{28}$$

By eliminating the closures with function symbols from (22)–(28), we obtain the following program $\mathsf{DD}(KB_3)$:

$$C(x) \leftarrow B(x) \tag{29}$$
$$D(x) \leftarrow R(x,y), C(y) \tag{30}$$
$$D(x) \leftarrow A(x) \tag{31}$$

Let us examine the role of each rule in $\mathsf{DD}(KB_3)$. The axiom $B \sqsubseteq C$ corresponds to (29), and the axiom $\exists R.C \sqsubseteq D$ corresponds to (30). The program $\mathsf{DD}(KB_3)$, however, does not contain an equivalent of the axiom $A \sqsubseteq \exists R.B$; hence, all rules in $\mathsf{DD}(KB_3)$ consider only the explicitly named individuals. To compensate for that, $\mathsf{DD}(KB_3)$ contains the rule (31), introduced in the saturation of $\Xi(KB_{\mathcal{T}})$ by $\mathcal{BS}^{+}_{DL}$. Instead of introducing, for each x in A, an R-successor y in B, propagating y to C, and then concluding that x is in D, the rule (31) derives in one step that all members of A are members of D, ensuring that $\mathsf{DD}(KB_3)$ and KB_3 entail the same ground facts.

Finally, let $KB_4 = \{\top \sqsubseteq \exists R.C,\ \top \sqsubseteq \leq 1\, R,\ \exists R.\exists R.\top \sqsubseteq D\}$. The translation into closures produces the following set $\Xi(KB_4)$:

$$\underline{R(x, f(x))} \tag{32}$$
$$\underline{C(f(x))} \tag{33}$$
$$\underline{\neg R(x, y_1)} \vee \underline{\neg R(x, y_2)} \vee y_1 \approx y_2 \tag{34}$$
$$\underline{\neg R(x,y)} \vee Q(x) \tag{35}$$
$$D(x) \vee \underline{\neg R(x,y)} \vee \neg Q(y) \tag{36}$$

By saturating $\Xi(KB_4)$ we obtain the following closures:

$$\underline{Q(x)} \quad \text{R(32;35)} \tag{37}$$
$$D(x) \vee \underline{\neg Q(f(x))} \quad \text{R(32;36)} \tag{38}$$
$$D(x) \quad \text{R(37;38)} \tag{39}$$

Because both binary literals in (34) are selected and there is only one closure with a positive binary R-literal, (32) and (34) do not participate in resolution; furthermore, hyperresolution inferences with a closure of type 7 and several copies of a side premise always produce a tautology. Consider now our translation into disjunctive datalog. Since $\Xi(KB)$ contains equality, we cannot simply eliminate all closures containing function symbols; rather, we must apply Definition 12. We obtain the following datalog program $\mathsf{DD}(KB_4)$:

$$R(x, x_f) \leftarrow S_f(x, x_f) \tag{40}$$

$$C(x_f) \leftarrow S_f(x, x_f) \tag{41}$$

$$y_1 \approx y_2 \leftarrow R(x, y_1), R(x, y_2) \tag{42}$$

$$Q(x) \leftarrow R(x, y) \tag{43}$$

$$D(x) \leftarrow R(x, y), Q(y) \tag{44}$$

$$Q(x) \leftarrow HU(x) \tag{45}$$

$$D(x) \leftarrow HU(x) \tag{46}$$

Translating the closure (38) produces the following rule:

$$D(x) \leftarrow Q(x_f), S_f(x, x_f) \tag{47}$$

This rule, however, can be obtained by resolving (40) and (44), so, by Definition 15, we need not include (47) into $\mathsf{DD}(KB_4)$. In contrast, the rules (45) and (46) cannot be obtained from other rules from $\mathsf{DD}(KB_4)$; for example, resolving (40) and (43) produces $Q(x) \leftarrow S_f(x, x_f)$ and not (45).

Let us now add the ABox containing only one assertion $R(a, b)$ to KB_4. To satisfy Definition 12, we additionally append the facts $S_f(a, a_f)$, $S_f(b, b_f)$, $HU(a)$, $HU(b)$, $HU(a_f)$, $HU(b_f)$ to $\mathsf{DD}(KB_4)$. We can then derive $C(b)$ in $\mathsf{DD}(KB_4)$ using the following inferences:

$$R(a, a_f) \tag{48}$$

$$C(a_f) \tag{49}$$

$$a_f \approx b \tag{50}$$

$$C(b) \tag{51}$$

The knowledge base from our example also implies $D(a)$; to derive this, we must consider the R-successors of a two steps away from a. The disjunctive datalog program derives $D(a)$ directly using the rule (46). Intuitively, this example shows that, in the presence of equality, we must take the immediate R-successors of a into account in order to derive all implied ground facts; however, the consequences of all longer paths are handled by the saturation step of the reduction.

Independence of the Reduction from the Query. Theorem 18 shows that $\mathsf{DD}(KB)$ is independent from the query if the query is a positive atomic concept or a role. Hence, $\mathsf{DD}(KB)$ can be computed once and used to answer any number of atomic queries.

For a nonatomic query concept C (even if C is a negated atomic concept), query answering can be reduced to entailment of positive ground facts by introducing a new name Q and adding the axiom $C \sqsubseteq Q$ to the TBox. Unfortunately, $\mathsf{DD}(KB \cup \{C \sqsubseteq Q\})$ depends on the query concept C. Intuitively, a complex concept C, even if used only in the query, introduces terminological knowledge which must be taken into account in the reduction.

$\mathsf{DD}(KB)$ cannot be used to answer conjunctive queries with nondistinguished variables—that is, existentially quantified variables that can be bound to any individual in the model; however, it can be used for answering conjunctive queries with only distinguished variables—that is, the existentially quantified variables that can be bound only to named individuals. Namely, $\mathsf{DD}(KB)$ implies a conjunction of ground atoms if and only if it implies each atom separately, and, by Theorem 18, the latter property can be checked using $\mathsf{DD}(KB)$. Similarly, $\mathsf{DD}(KB)$ can be used to check entailment of disjunctions of ground atoms.

Minimal vs. Arbitrary Models. Disjunctive datalog programs are usually interpreted under *minimal model* semantics: $P \models_c \alpha$ means that α is true in all minimal models of a program P, where minimality is defined w.r.t. set inclusion. Thus, disjunctive datalog implements a kind of *closed-world* semantics. In contrast, description logics assume the standard first-order, or *open-world* semantics: $KB \models \alpha$ means that α is true in all models of KB. It may come as a surprise that a logic based on arbitrary models can be embedded into a logic that considers only minimal models. This is possible because (*i*) the reduction algorithm produces only positive datalog programs (that is, programs without negation-as-failure), and (*ii*) first-order and minimal-model semantics coincide for certain types of questions.

For a positive datalog program P and a positive ground atom α, $P \models \alpha$ if and only if $P \models_c \alpha$. Namely, if α is true in each model of P, it is true in each minimal model of P as well and vice versa. Hence, for positive consequences, it is not important whether the semantics of P is defined w.r.t. minimal or w.r.t. general first-order models.

If α is a negative ground atom, the type of semantics matters, as we show in the following example. For $\alpha = \neg A(b)$ and $P = \{A(a)\}$, it is clear that $P \not\models \alpha$. Namely, $\neg A(b)$ is not explicitly derivable from the

facts in P: $M_1 = \{A(a), A(b)\}$ is a first-order model of P, and α is false in M_1. In contrast, P has exactly one minimal model $M_2 = \{A(a)\}$, and $\neg A(b)$ is obviously true in M_2, so $P \models_c \alpha$ (recall that the semantics of disjunctive datalog is defined only w.r.t. Herbrand models). In a similar vein, the choice of semantics affects concept subsumption.

It is therefore incorrect to check whether $KB \models \neg A(a)$ by checking whether $\mathsf{DD}(KB) \models_c \neg A(a)$. Instead, we must reduce the problem to entailment of positive ground facts: $KB \models \neg A(a)$ if and only if $\mathsf{DD}(KB \cup \{\neg A \sqsubseteq \mathit{NotA}\}) \models_c \mathit{NotA}(a)$, where NotA is a new concept.

Similarly, it is incorrect to check whether $KB \models C \sqsubseteq D$ by checking whether $\mathsf{DD}(KB) \models_c \forall x : [C(x) \rightarrow D(x)]$. Again, we must reduce the problem to entailment of positive ground facts. If C and D are atomic concepts, $KB \models C \sqsubseteq D$ if and only if $\mathsf{DD}(KB \cup \{C(a)\}) \models_c D(a)$, where a is a new individual not occurring in KB.

Complexity. Cautious query answering in a nonground positive disjunctive datalog program P can be performed in co-NExpTime [14]. Intuitively, the number of variables in a rule of P is linear in $|P|$, so the size of the ground program P^G, obtained by instantiating each rule from P with individuals in all possible ways, is exponential in $|P|$. Now satisfiability of P^G can be tested by guessing an interpretation (which is a nondeterministic exponential step), and by checking whether it is a model of P^G (which is an exponential step). Finally, query answering can be reduced to the complementary problem as usual. Actually, Eiter, Gottlob and Mannila give the complexity as co-NExpTime$^{\mathrm{NP}}$[14] because they consider a more general case of disjunctive datalog programs with negation-as-failure under stable model semantics. In such a case, it does not suffice to find an arbitrary model; one must additionally check if the model is minimal, which can be performed by an NP oracle.

Since $|\mathsf{DD}(KB)|$ is exponential in $|KB|$, one might get the impression that our algorithm increases complexity to (co-)2NExpTime. This is not the case because $\mathsf{DD}(KB)$ is of a restricted form. Namely, the number of variables in a rule is linear in $|KB|$, so a grounding of $\mathsf{DD}(KB)$ is exponential in $|KB|$. Furthermore, the predicates in $\mathsf{DD}(KB)$ are of limited arity, so an interpretation can be guessed in nondeterministic polynomial time, and checking whether it is a model can be done in exponential time. Finally, all interpretations can be examined in exponential time. To summarize, even though $|\mathsf{DD}(KB)|$ is exponential in $|KB|$, query answering can be performed in ExpTime because (i) the length of the rules in $\mathsf{DD}(KB)$ is polynomial in $|KB|$, and (ii) the arity of the literals is bounded.

Descriptive vs. Minimal Fixpoint Semantics. Our reduction preserves the so-called descriptive semantics. Namely, Nebel has observed that

knowledge bases containing terminological cycles are not definitorial [30]: for a fixed partial interpretation of primitive concepts, several interpretations of defined concepts may exist. In such a case, it might be reasonable to designate a particular interpretation as the intended one, with least and greatest fixpoint models being the obvious candidates. Nebel, however, argues that it is not clear which interpretation best matches the intuition, as choosing either of the fixpoint models has its drawbacks. Consequently, most description logic systems, as well as our approach, implement the descriptive semantics, which coincides with that of Definition 2.

Unique Name Assumption. The semantics of disjunctive datalog requires different symbols to be interpreted as different objects; however, this does not hold in $\mathcal{SHIQ}$. It may seem surprising that a logic without unique name assumption (UNA) can be embedded into a logic that strictly requires it.

To understand why our algorithms are correct, note that the UNA can be used to derive new consequences from a theory only if the theory employs equality. If a knowledge base uses neither number restrictions nor explicit individual equality statements, $\Xi(KB)$ does not contain the equality predicate, so equality reasoning is not needed. We might enforce UNA by appending $a \not\approx b$ for each $a \neq b$; however, these axioms would not participate in any inference, and are thus not needed in the first place. A model-theoretic explanation is given by the following claim, which holds for theories without any form of equality [15]: for each model I in which certain distinct constants are interpreted by the same objects, there is a model I' in which all constants are interpreted by distinct objects. To summarize, if KB does not employ explicit or implicit equality, we simply do not care whether either KB or $\mathsf{DD}(KB)$ employs UNA, as this does not change the entailed set of facts.

The situation changes for knowledge bases which require equality reasoning. Consider $KB = \{\top \sqsubseteq \leq 1\, R,\, R(a,b),\, R(a,c)\}$. Without UNA, KB is satisfiable and $KB \models b \approx c$. Namely, the first axiom requires R to be functional, so b and c must be interpreted as the same object. In contrast, with UNA, KB is unsatisfiable. Note that the program $\mathsf{DD}(KB)$ contains $y_1 \approx y_2 \leftarrow R(x,y_1), R(x,y_2)$. The equality occurring in the rule is quite different from the equality usually considered in disjunctive datalog. For example, in [14], equality atoms can only occur in rule bodies under negation-as-failure; such programs are interpreted under UNA, so an atom $\neg(x \approx y)$ actually checks whether x and y are bound to syntactically different individuals.

In contrast, we use equality not only in rule bodies, but also in rule heads. Hence, we can actually *derive* that two individuals are equal.

This is not directly supported in the implemented disjunctive datalog systems known to us; however, it can be simulated using the well-known encoding from [15] that treats $\approx$ as an ordinary predicate, explicitly states that it is reflexive, symmetric, transitive, and that it satisfies the usual equality replacement axioms. We thus obtain a program in which $\approx$ is just another predicate, which we can freely interpret with or without UNA, as discussed previously.

4. Data Complexity of Reasoning

Based on the reduction algorithm from Section 3, we now derive new *data complexity* results—that is, complexity under the assumption that the size of the ABox dominates the sizes of the TBox and RBox. In Section 4.1, we consider the case of full $\mathcal{SHIQ}$. In Section 4.2, we identify a fragment of $\mathcal{SHIQ}$ exhibiting polynomial data complexity.

4.1. Data Complexity for Full $\mathcal{SHIQ}$

The upper data complexity bound for $\mathcal{SHIQ}$ follows almost immediately from the reduction algorithm:

LEMMA 19 (Membership). *For KB an extensionally reduced $\mathcal{SHIQ}$ knowledge base, satisfiability of KB can be decided in nondeterministic polynomial time in $|KB_{\mathcal{A}}|$.*

Proof. Let c be the number of constants, f the number of function symbols, and s the number of facts in $\Xi(KB)$. By Definition 12, the number of constants in $\mathsf{DD}(KB)$ is bounded by $\ell_1 = c+cf$ (cf accounts for constants of the form a_f), and the number of facts in $\mathsf{DD}(KB)$ is bounded by $\ell_2 = s+c+2cf$ (c accounts for facts $HU(a)$, one cf accounts for facts $S_f(a, a_f)$, and the other cf accounts for facts $HU(a_f)$). All function symbols are introduced by skolemizing TBox concepts $\exists R.C$ and $\geq n\, R.C$. Since $|KB_{\mathcal{T}}|$ and $|KB_{\mathcal{R}}|$ are constant, f is also a constant, so ℓ_1 and ℓ_2 are linear in $|KB_{\mathcal{A}}|$.

Hence, $|\mathsf{DD}(KB)|$ can be exponential in $|KB|$ only because the nonground rules in $\mathsf{DD}(KB)$ are obtained from exponentially many closures of types 1–7. Since this does not include the ABox closures, the number of closures after saturation is exponential only in $|KB_{\mathcal{T}}|+|KB_{\mathcal{R}}|$. Since we assume that the latter is constant, both the number of rules in $\mathsf{DD}(KB)$ and their length are bounded by constants, so $|\mathsf{DD}(KB)|$ is polynomial in $|KB_{\mathcal{A}}|$, and can be computed from KB in time polynomial in $|KB_{\mathcal{A}}|$. Since KB and $\mathsf{DD}(KB)$ are equisatisfiable, the data complexity of checking satisfiability of KB follows from the data complexity

of checking satisfiability of $\mathsf{DD}(KB)$, which is in NP for disjunctive datalog programs without negation-as-failure [13]. □

Intuitively, TBox and RBox reasoning in $\mathcal{SHIQ}$ does not "interfere" with ABox reasoning; that is, all nonground consequences of KB can be computed without taking the ABox into account. Notice also that Lemma 19 holds even for binary coding of numbers.

The hardness of the satisfiability checking problem follows from [36, Lemma 4.2.7]. Actually, the lemma shows co-NP-hardness of instance checking by a reduction of satisfiability of 2-2-CNF propositional formulae. The reduction produces an extensionally reduced ABox and a single TBox axiom, so it is applicable to our case as well. Hence, we immediately obtain the following result:

THEOREM 20. *Let KB be an extensionally reduced knowledge base in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}$. Then, (*i*) deciding KB satisfiability is data complete for* NP *and (*ii*) deciding whether $KB \models (\neg)C(a)$ with $|C|$ bounded is data complete for co-*NP*.*

4.2. A Horn Fragment of $\mathcal{SHIQ}$

Horn logic is a well-known fragment of first-order logic in which formulae are restricted to clauses containing at most one positive literal. The main limitation of Horn logic is its inability to represent disjunctive information; however, its main benefit is the existence of practical refutation procedures. Data complexity of query answering in Horn logic without function symbols is P-complete [13], which makes it particularly appealing for practical usage. Following this idea, we now identify the Horn fragment of $\mathcal{SHIQ}$ that exhibits similar properties: in Horn-$\mathcal{SHIQ}$, the capability of representing disjunctive information is traded for polynomial data complexity of reasoning.

Horn-$\mathcal{SHIQ}$ allows only for axioms that do not require reasoning by case. Roughly speaking, only TBox axioms of the form $\bigsqcap C_i \sqsubseteq D$ are allowed, where C_i is of the form A, $\exists R.A$, or $\geq 1\, R.A$, and D is of the form A, $\bot$, $\exists R.A$, $\forall R.A$, $\geq n\, R.A$, or $\leq 1\, R$; additionally, we allow for role inclusion axioms and, in certain situations, transitivity axioms, as discussed after Definition 21. We call such axioms *simple Horn*.

Such a fragment of description logics is interesting because it can express many features of conceptual data models, such as the Entity-Relationship Model [11] or the Unified Modeling Language (UML); see [10] for more details. Horn-$\mathcal{SHIQ}$ is expressive enough to represent the following properties:

- inclusion of simple concepts (e.g., $\mathit{Woman} \sqsubseteq \mathit{Person}$)
- concept disjointness (e.g., $\mathit{Man} \sqcap \mathit{Woman} \sqsubseteq \bot$)
- domain restrictions (e.g., $\exists \mathit{husbandOf}.\top \sqsubseteq \mathit{Man}$)
- range restrictions (e.g., $\top \sqsubseteq \forall \mathit{husbandOf}.\mathit{Woman}$)
- functionality restriction (e.g., $\top \sqsubseteq \leq 1\, \mathit{husbandOf}$)
- participation constraints (e.g., $\mathit{Husband} \sqsubseteq \exists \mathit{husbandOf}.\mathit{Woman}$)
- role inclusions (e.g., $\mathit{husbandOf} \sqsubseteq \mathit{spouseOf}$)

Horn-$\mathcal{SHIQ}$ does not provide for the definition of covering constraints, such as $\mathit{Man} \sqcup \mathit{Woman} \equiv \mathit{Person}$.

Horn-$\mathcal{SHIQ}$ is a proper extension of DL-lite [9]—a description logic with LogSpace data complexity. Unlike DL-lite, Horn-$\mathcal{SHIQ}$ can express recursive axioms of the form $\exists R.C \sqsubseteq C$, which causes an increase in data complexity to polynomial time.

The definition of simple Horn axioms succinctly demonstrates the expressivity of the fragment, but it is too restricting in general. For example, the axiom $A_1 \sqcup A_2 \sqsubseteq \neg B$ is not simple Horn, but it is equivalent to simple Horn axioms $A_1 \sqcap B \sqsubseteq \bot$ and $A_2 \sqcap B \sqsubseteq \bot$. Similarly, an axiom $A \sqsubseteq \exists R.(\exists R.B)$ is not simple Horn, but it can be transformed into simple Horn axioms $A \sqsubseteq \exists R.Q$ and $Q \sqsubseteq \exists R.B$ by replacing $\exists R.B$ with a new name Q. To avoid dependency on such obvious syntactic transformations, we use the following, rather technical definition:

DEFINITION 21. *In Table V, we define two mutually recursive functions* pl^+ *and* pl^-*, where* $\mathsf{sgn}(0) = 0$ *and* $\mathsf{sgn}(n) = 1$ *for* $n > 0$*. For a concept* C *and a position* p *of a subconcept in* C*, let* $\mathsf{pl}(C,p) = \mathsf{pl}^+(C|_p)$ *if* $\mathsf{pol}(C,p) = 1$*, and let* $\mathsf{pl}(C,p) = \mathsf{pl}^-(C|_p)$ *if* $\mathsf{pol}(C,p) = -1$*.*

A concept C *is* Horn *if* $\mathsf{pl}(C,p) \leq 1$ *for each position* p *such that* $C|_p$ *is a concept (including the empty position* ϵ*). An extensionally reduced* $\mathcal{ALCHIQ}$ *knowledge base* KB *is* Horn *if, for each axiom* $C \sqsubseteq D \in KB$*, the concept* $\neg C \sqcup D$ *is Horn. An extensionally reduced* $\mathcal{SHIQ}$ *knowledge base* KB *is* Horn *if* $\Omega(KB)$ *is Horn.*

It is easy to see that, for a concept C without complex subconcepts, $\mathsf{pl}^+(C)$ gives the maximal number of positive literals in closures obtained by clausifying $\forall x : \pi_x(C)$. To clausify a concept C containing a complex subconcept at a position p, we should consider if $C|_p$ occurs in C under positive or negative polarity. For example, in $\neg(\neg A \sqcap \neg B)$, the concepts A and B occur effectively positively, and $\sqcap$ is effectively $\sqcup$. Hence, $\mathsf{pl}^+(C|_p)$ ($\mathsf{pl}^-(C|_p)$) counts the number of positive literals used to clausify $C|_p$, provided that $C|_p$ occurs in C under positive (negative) polarity. The function $\mathsf{sgn}(\cdot)$ takes into account that $C|_p$ will be replaced

Table V. Definitions of pl^+ and pl^-

D	$\mathsf{pl}^+(D)$	$\mathsf{pl}^-(D)$
$\top$	0	0
$\bot$	0	0
A	1	0
$\neg C$	$\mathsf{pl}^-(C)$	$\mathsf{pl}^+(C)$
$\bigsqcap C_i$	$\max_i\ \mathsf{sgn}(\mathsf{pl}^+(C_i))$	$\sum_i\ \mathsf{sgn}(\mathsf{pl}^-(C_i))$
$\bigsqcup C_i$	$\sum_i\ \mathsf{sgn}(\mathsf{pl}^+(C_i))$	$\max_i\ \mathsf{sgn}(\mathsf{pl}^-(C_i))$
$\exists R.C$	1	$\mathsf{sgn}(\mathsf{pl}^-(C))$
$\forall R.C$	$\mathsf{sgn}(\mathsf{pl}^+(C))$	1
$\geq n\, R.C$	1	$\frac{(n-1)n}{2} + n\,\mathsf{sgn}(\mathsf{pl}^+(C))$
$\leq n\, R.C$	$\frac{n(n+1)}{2} + (n+1)\mathsf{sgn}(\mathsf{pl}^-(C))$	1

in C by structural transformation with only one concept name, even if clausification of $C|_p$ produces more than one positive literal. For example, to clausify $C = \forall R.(D_1 \sqcup D_2)$, the structural transformation replaces $D_1 \sqcup D_2$ with a new concept name Q, yielding $C' = \forall R.Q$; then clausifying C' produces a closure with only one positive literal. Now a concept C is Horn if the maximal number of positive literals obtained by clausifying subconcepts of C is at most one.

If a concept C has a complex subconcept at position p, care has to be taken in introducing a new name for $C|_p$. Consider the Horn concept $C = \forall R.D_1 \sqcup \forall R.\neg D_2$. In applying structural transformation, one might replace $\forall R.D_1$ and $\forall R.\neg D_2$ with new concept names Q_1 and Q_2, respectively, yielding concepts $\neg Q_1 \sqcup \forall R.D_1$, $\neg Q_2 \sqcup \forall R.\neg D_2$, and $Q_1 \sqcup Q_2$. The problem with this approach is that a Horn concept C was reduced to a non-Horn concept $Q_1 \sqcup Q_2$, so the structural transformation destroyed Horn-ness. To remedy this, we modify the structural transformation to replace each $C|_p$ with a literal concept α such that clausifying α and $C|_p$ requires the same number of positive literals. In the above example, this would mean that $\forall R.D_1$ should be replaced with Q_1, but $\forall R.\neg D_2$ should be replaced with $\neg Q_2$, yielding concepts $\neg Q_1 \sqcup \forall R.D_1$, $Q_2 \sqcup \forall R.\neg D_2$, and $Q_1 \sqcup \neg Q_2$, all of which are Horn.

Although transitivity axioms are translated by π into Horn closures, the algorithm from Section 3.2 replaces them with axioms of the form $\forall R.C \sqsubseteq \forall S.(\forall S.C)$. Now $\mathsf{pl}^+(\exists R.\neg C \sqcup \forall S.(\forall S.C)) = 1 + \mathsf{pl}^+(C)$, so if $\mathsf{pl}^+(C) > 0$, $\Omega(KB)$ is not a Horn knowledge base. Hence, the presence of transitivity axioms can make a knowledge base non-Horn.

DEFINITION 22. *The* Horn-compatible *structural transformation is as in Definition 3, with the following difference to* $\mathsf{Def}(C)$*, where* $\alpha = Q$ *if* $\mathsf{pl}(C,p) > 0$*, and* $\alpha = \neg Q$ *if* $\mathsf{pl}(C,p) = 0$*, for* Q *a new atomic concept and* $\neg(\neg Q) = Q$*:*

$$\mathsf{Def}(C) = \begin{cases} \{\neg\alpha \sqcup C|_p\} \cup \mathsf{Def}(C[\alpha]_p) & \text{if } \mathsf{pol}(C,p) = 1 \\ \{\neg\alpha \sqcup \neg C|_p\} \cup \mathsf{Def}(C[\neg\alpha]_p) & \text{if } \mathsf{pol}(C,p) = -1 \end{cases}$$

By [32], $\forall x : \pi_x(C)$ and $\bigwedge_{D \in \mathsf{Def}(C)} \forall x : \pi_x(D)$ are equisatisfiable, so $\Xi(KB)$ and $\pi(KB)$ are equisatisfiable as well.

LEMMA 23. *For a Horn-*$\mathcal{SHIQ}$ *knowledge base* KB*, each closure from* $\Xi(KB)$ *contains at most one positive literal.*

Proof. We first show the following property (*): for a Horn concept C, all concepts in $\mathsf{Def}(C)$ are Horn concepts. The proof is by induction on the recursion depth. The induction base for $\Lambda(C) = \emptyset$ is obvious. Consider an application of $\mathsf{Def}(C)$, where C is a Horn concept and p a position of a subconcept of C, such that $C|_p$ is not a literal concept and, for each position q below p, $C|_q$ is a literal concept. In all cases, we have $\mathsf{pl}^+(\alpha) = \mathsf{pl}(C,p)$ and $\mathsf{pl}^+(\neg\alpha) = 1 - \mathsf{pl}(C,p)$. If $\mathsf{pol}(C,p) = 1$, then $\mathsf{pl}^+(\neg\alpha \sqcup C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}^+(C|_p) = \mathsf{pl}^+(\neg\alpha) + \mathsf{pl}(C,p) = 1$; furthermore, $\mathsf{pl}(C,p) = \mathsf{pl}(C[\alpha]_p, p)$, so $C[\alpha]_p$ is Horn. Similar considerations hold for $\mathsf{pol}(C,p) = -1$. Hence, Def decomposes a Horn concept C into two simpler Horn concepts, so (*) holds.

For $D \in \mathsf{Def}(C)$, one can see that $\mathsf{pl}^+(D)$ gives the maximal number of positive literals occurring in $\mathsf{Cls}(\forall x : \pi_x(D))$. Thus, if C is a Horn concept, all closures from $\mathsf{Cls}(C)$ contain at most one positive literal. Finally, the closures of $\Omega(KB)$ obtained by translating the RBox and the ABox also contain at most one positive literal. □

LEMMA 24. *If all premises of an inference by* $\mathcal{BS}^+_{DL}$ *contain at most one positive literal, then inference conclusions also contain at most one positive literal.*

Proof. In ordered hyperresolution and positive or negative superposition, each side premise participates in an inference on the positive literal, which does not occur in the conclusion. Hence, the number of positive literals in the conclusion is equal to the number of positive literals in the main premise. Furthermore, reflexivity resolution only reduces the number of negative literals in a closure, and equality factoring is never applicable to a closure with only one positive literal. Finally, a closure participating in a decomposition inference contains the single positive literal $R(t, f(t))$, so both resulting closures have exactly one positive literal. □

LEMMA 25 (Hardness). *For a Horn $\mathcal{ALC}$ knowledge base KB, instance checking w.r.t. KB is* P*-hard in* $|KB_{\mathcal{A}}|$.

Proof. The proof is by a reduction from the well-known Boolean circuit value problem [33]. A *Boolean circuit* C is a graph (G, δ, E) defined as follows: (i) $G = \{\gamma_1, ..., \gamma_n\}$ is the set of nodes, also called *gates*; (ii) $\delta : G \rightarrow \{T, F, \wedge, \vee, \neg\}$ is a function assigning a label to each gate; and (iii) $E \subseteq G \times G$ is the acyclic set of edges such that the in-degree of gates labeled with T or F is zero, of gates labeled with $\neg$ is one, and of gates labeled with $\wedge$ or $\vee$ is two. A valuation $\mu : G \rightarrow \{T, F\}$ over gates of C is defined according to standard truth tables for Boolean connectives, and the *value* of C is defined as $\mu(C) = \mu(\gamma_n)$. For an arbitrary circuit C, checking whether $\mu(C) = T$ is P-complete [33].

For a Boolean circuit C, we construct the knowledge base KB_C in which each gate γ_i corresponds to an individual γ_i. Let T and F be concept names, and let not, and_1, and_2, or_1, and or_2 be role names. We convert C into ABox assertions as follows: (i) add $not(\gamma, \gamma_1)$ for all nodes γ, γ_1 such that $\delta(\gamma) = \neg$ and $(\gamma_1, \gamma) \in E$; (ii) add $and_1(\gamma, \gamma_1)$ and $and_2(\gamma, \gamma_2)$ for all nodes $\gamma, \gamma_1, \gamma_2$ such that $\delta(\gamma) = \wedge$, $(\gamma_1, \gamma) \in E$, and $(\gamma_2, \gamma) \in E$; (iii) add $or_1(\gamma, \gamma_1)$ and $or_2(\gamma, \gamma_2)$ for all nodes $\gamma, \gamma_1, \gamma_2$ such that $\delta(\gamma) = \vee$, $(\gamma_1, \gamma) \in E$, and $(\gamma_2, \gamma) \in E$; (iv) add $T(\gamma)$ for each node γ such that $\delta(\gamma) = T$; and (v) add $F(\gamma)$ for each node γ such that $\delta(\gamma) = F$. The TBox of KB_C contains the following axioms:

$$\begin{array}{rr}
\exists not.T \sqsubseteq F & T \sqcap F \sqsubseteq \bot \\
\exists not.F \sqsubseteq T & \\
\exists and_1.T \sqcap \exists and_2.T \sqsubseteq T & \exists or_1.T \sqcap \exists or_2.T \sqsubseteq T \\
\exists and_1.T \sqcap \exists and_2.F \sqsubseteq F & \exists or_1.T \sqcap \exists or_2.F \sqsubseteq T \\
\exists and_1.F \sqcap \exists and_2.T \sqsubseteq F & \exists or_1.F \sqcap \exists or_2.T \sqsubseteq T \\
\exists and_1.F \sqcap \exists and_2.F \sqsubseteq F & \exists or_1.F \sqcap \exists or_2.F \sqsubseteq F
\end{array}$$

The TBox axioms of KB_C obviously implement the standard semantics of propositional connectives, so, for each gate γ, $\mu(\gamma) = T$ ($\mu(\gamma) = F$) if and only if $KB_C \models T(\gamma)$ ($KB_C \models F(\gamma)$). The size of the TBox of KB_C is constant, the size of the ABox of KB_C is linear in the size of C, and KB_C is a Horn knowledge base, so the claim follows. □

By Lemma 23 and 24, if KB is a Horn-$\mathcal{SHIQ}$ knowledge base, then $\mathsf{DD}(KB)$ is a Horn program. This implies the following result:

THEOREM 26. *For KB an extensionally reduced Horn knowledge base in any logic between $\mathcal{ALC}$ and $\mathcal{SHIQ}$, deciding KB (un)satisfiability, and deciding whether $KB \models (\neg)C(a)$ with $|C|$ bounded, is* P*-complete in* $|KB_{\mathcal{A}}|$.

Proof. Membership in P is a consequence of the fact that $\mathsf{DD}(KB)$ is a Horn program, whose satisfiability can be checked in polynomial time [13]. Hence, the claim of this theorem follows from Lemma 25. □

5. Related Work

We now discuss related approaches to reasoning in description logics via rule-based formalisms. Our work was largely motivated by Grosof, Horrocks, Decker, and Volz, who have investigated a decidable intersection of description logic and logic programming [16]. In particular, the authors identify the description logic constructs that can be straightforwardly encoded and executed using existing logic programming systems. Thus, the DL component allows only existential quantifiers to occur under negative, and universal quantifiers to occur under positive polarity. The authors present an operator for translating a description logic knowledge base into a logic program. Our approach is a significant extension since we handle the DL $\mathcal{SHIQ}$, which requires a more complex reduction.

Heymans and Vermeir showed how to convert $\mathcal{SHIQ}^*$ knowledge bases into *conceptual logic programs* (CLP) [19]. CLPs generalize the good properties of description logic to the framework of answer set programming. Apart from the usual constructs, $\mathcal{SHIQ}^*$ supports the transitive closure of roles. Although the presented transformation preserves the semantics of the knowledge base, the resulting answer set program is not safe. Hence, its grounding is infinite, so the program cannot be evaluated using existing answer set solvers. The problem of decidable reasoning for CLPs is addressed by an automata-based technique. In contrast, our transformation produces a safe program with a finite grounding, so decidability of reasoning is guaranteed already by the transformation.

An approach for deciding satisfiability of DL concepts using answer set programming was presented in [37]. This work, however, does not consider general concept inclusion axioms. Another approach for reducing description logic knowledge bases to answer set programming was developed by Alsaç and Baral [2]. To deal with existential quantification, this approach uses function symbols. Thus, the Herbrand universe of the programs obtained by the reduction is infinite, so existing answer set solvers cannot be used for reasoning. In fact, decidability is not considered at all.

6. Conclusion

We have presented an algorithm for reducing a $\mathcal{SHIQ}$ knowledge base to a disjunctive datalog program which entails the same set of ground facts as the original knowledge base. Thus, DL query answering can be implemented using various optimizations that were developed for disjunctive datalog, such as join order optimizations or the magic sets transformation. The latter has been shown to dramatically improve the evaluation of disjunctive datalog programs. Our experiments, reported in [29], indeed show significant performance improvements over existing DL systems in answering queries over knowledge bases with large ABoxes but modest TBoxes.

Furthermore, our approach allows us to derive tight data complexity bounds—that is, the complexity under the assumption that the TBox is stable, whereas the ABox is varying and is possibly very large. We show that checking satisfiability of a $\mathcal{SHIQ}$ knowledge base is NP-complete, and that unsatisfiability and instance checking are co-NP-complete in the size of the ABox. Additionally, we identify Horn-$\mathcal{SHIQ}$, a fragment of $\mathcal{SHIQ}$ which, analogously to Horn logic, does not allow for disjunctive knowledge, and for which the basic reasoning problems are P-complete in the size of the ABox.

For our future work, the main challenge lies in extending the algorithm to $\mathcal{SHOIQ}$. In [27] we presented a resolution-based decision procedure for this logic; however, it is currently not clear whether this result can be used to extend the reduction algorithm presented here.

References

1. Abiteboul, S., R. Hull, and V. Vianu: 1995, *Foundations of Databases*. Addison Wesley.
2. Alsaç, G. and C. Baral: 2002, 'Reasoning in description logics using declarative logic programming'. Technical report, Arizona State University, Arizona, USA. http://www.public.asu.edu/ cbaral/papers/descr-logic-aaai2.pdf.
3. Baader, F., D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider (eds.): 2003, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.
4. Baader, F. and T. Nipkow: 1998, *Term Rewriting and All That*. Cambridge University Press.
5. Baaz, M., U. Egly, and A. Leitsch: 2001, 'Normal Form Transformations'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. I. Elsevier Science, Chapt. 5, pp. 273–333.
6. Bachmair, L., H. Ganzinger, C. Lynch, and W. Snyder: 1995, 'Basic Paramodulation'. *Information and Computation* **121**(2), 172–192.

7. Beeri, C. and R. Ramakrishnan: 1987, 'On the power of magic'. In: *Proc. of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '87)*. San Diego, CA, USA, pp. 269–283.
8. Borgida, A.: 1996, 'On the Relative Expressiveness of Description Logics and Predicate Logics'. *Artificial Intelligence* **82**(1–2), 353–367.
9. Calvanese, D., G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati: 2006, 'Data Complexity of Query Answering in Description Logics'. In: P. Doherty, J. Mylopoulos, and C. A. Welty (eds.): *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*. Lake District, United Kingdom, pp. 260–270.
10. Calvanese, D., M. Lenzerini, and D. Nardi: 1998, 'Description Logics for Conceptual Data Modeling'. In: J. Chomicki and G. Saake (eds.): *Logics for Databases and Information Systems*. Kluwer, Chapt. 8, pp. 229–263.
11. Chen, P. P.: 1976, 'The Entity-Relationship Model - Toward a Unified View of Data'. *ACM Transactions on Database Systems* **1**(1), 9–36.
12. Cumbo, C., W. Faber, G. Greco, and N. Leone: 2004, 'Enhancing the Magic-Set Method for Disjunctive Datalog Programs'. In: B. Demoen and V. Lifschitz (eds.): *Proc. of the 20th Int. Conf. on Logic Programming (ICLP 2004)*, Vol. 3132 of *LNCS*. Saint-Malo, France, pp. 371–385.
13. Dantsin, E., T. Eiter, G. Gottlob, and A. Voronkov: 2001, 'Complexity and expressive power of logic programming'. *ACM Computing Surveys* **33**(3), 374–425.
14. Eiter, T., G. Gottlob, and H. Mannila: 1997, 'Disjunctive Datalog'. *ACM Transactions on Database Systems* **22**(3), 364–418.
15. Fitting, M.: 1996, *First-Order Logic and Automated Theorem Proving, 2nd Edition*, Texts in Computer Science. Springer.
16. Grosof, B. N., I. Horrocks, R. Volz, and S. Decker: 2003, 'Description Logic Programs: Combining Logic Programs with Description Logic'. In: *Proc. of the 12th Int. World Wide Web Conference (WWW 2003)*. Budapest, Hungary, pp. 48–57.
17. Haarslev, V. and R. Möller: 2001, 'RACER System Description'. In: R. Goré, A. Leitsch, and T. Nipkow (eds.): *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, Vol. 2083 of *LNAI*. Siena, Italy, pp. 701–706.
18. Haarslev, V., R. Möller, and A.-Y. Turhan: 2001, 'Exploiting Pseudo Models for TBox and ABox Reasoning in Expressive Description Logics'. In: R. Goré, A. Leitsch, and T. Nipkow (eds.): *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, Vol. 2083 of *LNAI*. Siena, Italy, pp. 61–75.
19. Heymans, S. and D. Vermeir: 2003, 'Integrating Semantic Web Reasoning and Answer Set Programming'. In: M. D. Vos and A. Provetti (eds.): *Proc. of the 2nd Int. Workshop on Answer Set Programming, Advances in Theory and Implementation (ASP'03)*, Vol. 78 of *CEUR Workshop Proceedings*. Messina, Italy, pp. 194–208.
20. Horrocks, I., P. F. Patel-Schneider, and F. van Harmelen: 2003, 'From SHIQ and RDF to OWL: the making of a Web Ontology Language'. *Journal of Web Semantics* **1**(1), 7–26.
21. Horrocks, I., U. Sattler, and S. Tobies: 2000, 'Practical Reasoning for Very Expressive Description Logics'. *Logic Journal of the IGPL* **8**(3), 239–263.
22. Hustadt, U., B. Motik, and U. Sattler: 2004, 'Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs'. In: D. Dubois, C. A. Welty, and M.-A. Williams (eds.): *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*. Whistler, Canada, pp. 152–162.

23. Hustadt, U., B. Motik, and U. Sattler: 2005a, 'A Decomposition Rule for Decision Procedures by Resolution-based Calculi'. In: F. Baader and A. Voronkov (eds.): *Proc. of the 11th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2004)*, Vol. 3452 of *LNAI*. Montevideo, Uruguay, pp. 21–35.
24. Hustadt, U., B. Motik, and U. Sattler: 2005b, 'Data Complexity of Reasoning in Very Expressive Description Logics'. In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Edinburgh, UK, pp. 466–471.
25. Hustadt, U. and R. A. Schmidt: 1999, 'On the Relation of Resolution and Tableaux Proof Systems for Description Logics'. In: D. Thomas (ed.): *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI '99)*. Stockholm, Sweden, pp. 110–115.
26. Kazakov, Y. and H. de Nivelle: 2004, 'A Resolution Decision Procedure for the Guarded Fragment with Transitive Guards'. In: D. Basin and M. Rusinowitch (eds.): *Proc. of 2nd Int. Joint Conf. on Automated Reasoning (IJCAR 2004)*, Vol. 3097 of *LNAI*. Cork, Ireland, pp. 122–136.
27. Kazakov, Y. and B. Motik: 2006, 'A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$'. In: U. Furbach, J. Harrison, and N. Shankar (eds.): *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *LNAI*. Seattle, WA, USA, pp. 662–667.
28. Motik, B.: 2006, 'Reasoning in Description Logics using Resolution and Deductive Databases'. Ph.D. thesis, Univesität Karlsruhe, Germany.
29. Motik, B. and U. Sattler: 2006, 'A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes'. In: M. Hermann and A. Voronkov (eds.): *Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*. Phnom Penh, Cambodia. Accepted for publication.
30. Nebel, B.: 1991, 'Terminological Cycles: Semantics and Computational Properties'. In: J. F. Sowa (ed.): *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. San Mateo, CA, USA: Morgan Kaufmann Publishers, pp. 331–361.
31. Nieuwenhuis, R. and A. Rubio: 1995, 'Theorem Proving with Ordering and Equality Constrained Clauses'. *Journal of Symbolic Computation* **19**(4), 312–351.
32. Nonnengart, A. and C. Weidenbach: 2001, 'Computing Small Clause Normal Forms'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. I. Elsevier Science, Chapt. 6, pp. 335–367.
33. Papadimitriou, C. H.: 1993, *Computational Complexity*. Addison Wesley.
34. Parsia, B. and E. Sirin, 'Pellet: An OWL-DL Reasoner'. Poster, In Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 7–11, 2004.
35. Plaisted, D. A. and S. Greenbaum: 1986, 'A Structure-Preserving Clause Form Translation'. *Journal of Symbolic Logic and Computation* **2**(3), 293–304.
36. Schaerf, A.: 1994, 'Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues'. Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Italy.
37. Swift, T.: 2004, 'Deduction in Ontologies via ASP'. In: V. Lifschitz and I. Niemelä (eds.): *Proc. of the 7th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2004)*, Vol. 2923 of *LNCS*. Fort Lauderdale, FL, USA, pp. 275–288.

38. Tobies, S.: 2001, 'Complexity Results and Practical Algorithms for Logics in Knowledge Representation'. Ph.D. thesis, RWTH Aachen, Germany.
39. Tsarkov, D. and I. Horrocks: 2006, 'FaCT++ Description Logic Reasoner: System Description'. In: *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *LNAI*. Seattle, WA, USA, pp. 292–297.