A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$ *

Yevgeny Kazakov
Computing Laboratory, University of Oxford, UK

Boris Motik
Computing Laboratory, University of Oxford, UK

Abstract. We present a resolution-based decision procedure for the description logic $\mathcal{SHOIQ}$—the logic underlying the Semantic Web ontology language OWL-DL. Our procedure is goal-oriented, and it naturally extends a similar procedure for $\mathcal{SHIQ}$, which has proven itself in practice. Extending this procedure to $\mathcal{SHOIQ}$ using existing techniques is not straightforward because of nominals, number restrictions, and inverse roles—a combination known to cause termination problems. We overcome this difficulty by using the basic superposition calculus extended with custom simplification rules.

Keywords: Description Logics, Resolution Decision Procedures, Nominals

1. Introduction

Description logics (DLs) are a family of knowledge representation formalisms [2] with applications in diverse fields of computer science such as information integration [2, Chapter 16] [33, 9, 19], software engineering [2, Chapter 11], and conceptual modeling [2, Chapter 10] [11]. Furthermore, DLs provide the logical foundation for the Web Ontology Language (OWL) [40]—the language standardized by the W3C for building Semantic Web ontologies. The OWL DL variant of OWL is closely related to the DL $\mathcal{SHOIQ}$. To support the applications of DLs, several reasoning algorithms were developed and implemented.[1]

It is well known that $\mathcal{SHOIQ}$ can be embedded in $\mathcal{C}^2$ [46]—the two-variable fragment of first-order logic with counting quantifiers—and that $\mathcal{C}^2$ can be decided in NExpTime [39] (this was recently shown to hold for binary coding of numbers [43] as well). Although these results imply decidability of $\mathcal{SHOIQ}$, they are unlikely to be suitable for practical purposes. Specifically, the algorithms from [39, 43] check satisfiability of a $\mathcal{C}^2$ formula by "blindly" guessing a representation of an interpretation of the formula and then verifying whether this interpretation is a model of the formula. Since this representation can be

---

exponentially large in the size of the formula signature, these algorithms are impractical.

A practical decision algorithm should be goal-oriented: it should use the formula structure, and not only the formula signature in the search for a model. Currently, the state-of-the-art practical reasoning algorithms for DLs are tableau algorithms [2]. Tableau algorithms have been derived for DLs such as $\mathcal{SHIQ}$ [24], but extending these results to $\mathcal{SHOIQ}$—the extension of $\mathcal{SHIQ}$ with singleton concepts called *nominals*—has proved to be a nontrivial task. The reason rests with the intricate interaction between inverse roles, number restrictions, and nominals. Only recently, a goal-directed tableau-based procedure was presented in [23]; it uses an additional rule that nondeterministically creates nominals to ensure termination.

$\mathcal{SHOIQ}$ is a complex DL, so it is not obvious which reasoning algorithm is best in practice. Rather, it makes sense to compare different algorithms and to identify which ones are suitable for which types of problems. The state-of-the-art techniques for first-order theorem proving are based on resolution, so it is natural to try using them for DL reasoning. Resolution-based decision procedures have been developed for many different DLs (see Section 3 for an overview). In particular, a resolution-based algorithm for reducing a $\mathcal{SHIQ}$ knowledge base to a disjunctive datalog program has been presented in [25]. This algorithm has been implemented in the reasoning system KAON2[2] and has proved itself on problems with many individual assertions [35].

Here, we continue this line of research and present a resolution-based decision procedure for the DL $\mathcal{SHOIQ}$. In building our procedure, we encountered problems analogous to those arising in the tableau algorithm from [23]. Namely, the combination of nominals, inverse roles, and number restrictions can cause resolution to derive clauses of unbounded size, thus preventing termination. We solve these problems by using basic superposition [6, 36], which we extend with novel *simplification* rules. These rules replace complex clauses with simpler ones without compromising soundness or completeness, while ensuring a bound on the number of derivable clauses.

This paper is organized as follows. After presenting the basic definitions in Section 2, we discuss the problems and the solutions involved in deriving the decision procedure in Section 3. We split our procedure into two phases: in Section 4 we present the preprocessing phase, and in Section 5 we present the saturation phase of the algorithm. Before concluding, we discuss interesting aspects of our algorithm in Section 6.

---

[2] `http://kaon2.semanticweb.org/`

2. Preliminaries

2.1. Basic Notions of First-Order Logic

We use the standard notions of first-order logic with equality and *counting quantifiers* $\exists^{\geq n}$ and $\exists^{\leq n}$. Throughout this paper, we use x, y, and z for variables; a, b, c, and d for constants; s, t, u, v, and w for terms; and f, g, and h for function symbols (possibly with subscripts or superscripts). Terms, atoms, and formulae that do not contain variables are said to be *ground*. We use the infix notation for the equality predicate symbol $\approx$ and do not distinguish the *equational atoms* $t \approx s$ and $s \approx t$ or their negations $t \not\approx s$ and $s \not\approx t$. Counting quantifiers can be expressed by using ordinary quantifiers and equality as follows.

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

$$(2)\, \exists^{\leq n} x.\varphi(x,\overline{y}) \;=\; \forall x_1,\ldots,x_{n+1}. \left[\bigwedge_{i=1}^{n+1} \varphi(x_i,\overline{y}) \to \bigvee_{1 \leq i < j \leq n+1} x_i \approx x_j \right]$$

In Section 3.1.2, we present an alternative encoding of counting quantifiers, which is more suitable for our decision procedure.

A *substitution* σ is an assignment of variables to terms, which we write as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. With $\mathsf{MGU}(s,t)$ we denote the *most general unifier* of s and t.

A *position* p is a finite sequence of integers, written as $i_1.i_2 \ldots i_n$. The empty position is denoted with ϵ. Let t be a term of the form $t = f(t_1, \ldots, t_i, \ldots, t_n)$; when n is zero, f is a constant. The subterm of t at position p, written $t|_p$, is defined as $t|_\epsilon = t$ and $t|_{i.p} = t_i|_p$. The replacement of a subterm of t at position p with a term s, written $t[s]_p$, is defined as $t[s]_\epsilon = s$ and $t[s]_{i.p} = f(t_1, \ldots, t_i[s]_p, \ldots, t_n)$.

A *strict (partial) ordering* $\succ$ on a set D is a transitive irreflexive binary relation on D. The *reflexive closure* of $\succ$ is denoted by $\succeq$. An ordering $\succ$ is *total* if, for each $d_1 \neq d_2$ from D, either $d_1 \succ d_2$ or $d_2 \succ d_1$. An ordering $\succ$ is *well-founded* if there is no infinite sequence $d_1 \succ d_2 \succ \cdots$. An ordering $\succ$ on terms is a *reduction ordering* if $s \succ t$ implies $u[s]_p \succ u[t]_p$ for all terms s, t, and u and all positions p.

A *multiset* M over a set D is a function $M : D \to \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of all nonnegative integers. The size of M is defined as $|M| = \Sigma_{d \in D} M(d)$; M is *finite* if $|M|$ is finite. A multiset M is a *sub-multiset* of N, written $M \subseteq N$, if $M(x) \leq N(x)$ for every $x \in D$; the *union* of M and N is defined as $(M \cup N)(x) = M(x) + N(x)$. Given a strict ordering $\succ$ on D, the *multiset extension* of $\succ$ to finite multisets

over D, written $\succ_{mul}$, is defined as follows: $M \succ_{mul} N$ if (i) $M \neq N$, and (ii) for every $x \in D$ with $N(x) > M(x)$ there exists some $y \succ x$ such that $M(y) > N(y)$.

A *literal* is an atom A or a negated atom $\neg A$. The literals A and $\neg A$ are said to be *complementary*. For a literal L, with $\overline{L}$ we denote the literal complementary to L. A *clause* is a finite multiset of literals and is written as $C = L_1 \vee \ldots \vee L_n$; it is semantically equivalent to $\forall \overline{x}.C$, where $\overline{x}$ is the vector of the free variables of C. If $n = 1$, C is a *unit clause*; if $n = 0$, C is the *empty clause* and is written as $\square$. A ground clause C' is a *ground instance* of a clause C if a substitution σ exists such that $C' = C\sigma$. A *Herbrand interpretation* I is a set of ground atoms. A ground literal L is satisfied in a Herbrand interpretation I, written $I \models L$, if either $L = A$ and $A \in I$, or $L = \neg A$ and $A \notin I$. A clause C is satisfied in I if, for every ground instance C' of C, there is a literal $L \in C'$ such that $I \models L$.

2.2. Description Logic $\mathcal{SHOIQ}$

In this section we introduce the syntax and the semantics of the description logic $\mathcal{SHOIQ}$. A $\mathcal{SHOIQ}$ *signature* is a triple $\Sigma = (N_R, N_C, N_I)$, where N_R is a set of *role names*, N_C is a set of *concept names*, and N_I is a set of *individuals*. A *role* is either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$; for each $R \in N_R$, we set $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$.

An *RBox* $KB_{\mathcal{R}}$ is a finite set of role *inclusion axioms* $R \sqsubseteq S$ and *transitivity axioms* $\mathsf{Trans}(R)$, where R and S are 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 S is *simple* if no role R exists such that $R \sqsubseteq^* S$ and either $\mathsf{Trans}(R) \in KB_{\mathcal{R}}$ or $\mathsf{Trans}(\mathsf{Inv}(R)) \in KB_{\mathcal{R}}$.

The set of *concepts* is the smallest set containing $\top, \bot, A, \neg C, C \sqcap D$, $C \sqcup D$, $\{a\}$, $\exists R.C$, $\forall R.C$, $\geqslant n\, S.C$, and $\leqslant n\, S.C$, where A is a concept name, C and D are concepts, R is a role, S is a simple role, a is an individual, and n is a nonnegative integer.

A *TBox* is a finite set $KB_{\mathcal{T}}$ of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are concepts. An *ABox* is a finite set $KB_{\mathcal{A}}$ of axioms $C(a)$, $R(a, b)$, and (in)equalities $a \approx b$ and $a \not\approx b$, for $a, b \in N_I$. A $\mathcal{SHOIQ}$ knowledge base is a triple $KB = (KB_{\mathcal{R}}, KB_{\mathcal{T}}, KB_{\mathcal{A}})$. With $|KB|$ we denote the number of symbols in KB assuming that the numbers in number restrictions are written in unary coding.

Usually, KB is interpreted by using a direct model-theoretic semantics (e.g., see [2, Chapter 2]). Since our reasoning algorithms are based on resolution, in this paper we define the semantics of $\mathcal{SHOIQ}$ by translating KB into a formula $\pi(KB)$ of first-order logic with equality and counting quantifiers; the translation operator π is defined in Table

Table I. Semantics of $\mathcal{SHOIQ}$ by Mapping to FOL

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

I. In the translation, concept names correspond to unary predicate symbols and roles correspond to binary predicate symbols (for an inverse role R^-, we take R^- to be a name of a binary predicate symbol). It is well known that the translational semantics is equivalent to the direct model-theoretic one [10]. The main reasoning problem for $\mathcal{SHOIQ}$ is *checking satisfiability* of KB, which corresponds to checking first-order satisfiability of $\pi(KB)$. All other commonly considered inference problems, such as *concept satisfiability*, *concept subsumption*, and *instance checking*, can be reduced to satisfiability using the well-known transformations [2, Chapter 2].

The DL $\mathcal{ALCHOIQ}$ is obtained from the DL $\mathcal{SHOIQ}$ by disallowing the transitivity axioms. The DL $\mathcal{SHIQ}$ is obtained from the DL $\mathcal{SHOIQ}$ by disallowing nominals. With $\mathsf{nnf}(C)$ we denote the *negation-normal form* of C—that is, the concept equivalent to C in which negation occurs only in front of concept names. It is well known that $\mathsf{nnf}(C)$ can be computed in time polynomial in C [2, Chapter 2].

2.3. Basic Superposition Calculus

We briefly recapitulate the relevant notions of resolution theorem proving [5, 37]. In equational theorem proving, one usually assumes that equality is the only predicate symbol. Hence, each literal $P(t_1, \ldots, t_n)$ where P is not equality is encoded as $P(t_1, \ldots, t_n) \approx \mathtt{tt}$, where $\mathtt{tt}$ is a new propositional symbol. If P and $\mathtt{tt}$ are of a sort different from the sort of the 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.

Basic superposition ($\mathcal{BS}$) [6, 36] is a calculus optimized for equational theorem proving. The inference rules of $\mathcal{BS}$ work with *closures*, which consist of a *skeleton* clause C and a *substitution* σ. A closure, written as $C \cdot \sigma$, semantically represents the clause $C\sigma$. A closure $C \cdot \sigma$ is *ground*, *unit*, or *empty* if $C\sigma$ is ground, unit, or empty, respectively. If σ is identity, $C \cdot \sigma$ is often written as C; furthermore, $(C_1 \vee C_2) \cdot \sigma$ is often written as $C_1 \cdot \sigma \vee C_2 \cdot \sigma$. Moreover, we sometimes use (closure) terms $t \cdot \sigma$ and (closure) literals $L \cdot \sigma$.

A closure can be written by using the *abbreviated form* as a clause $C\sigma$ in which the positions of the variables of C are enclosed in square brackets. For example, the following are two forms of the same closure:

$$(P(x) \vee z \approx b) \cdot \{x \mapsto f(y), z \mapsto g(b)\} \equiv P([f(y)]) \vee [g(b)] \approx b.$$

The positions that correspond to terms enclosed in square brackets are called *marked positions*, and the positions at or below marked positions are called *substitution positions*. We do not distinguish closures that have the same abbreviated form. For example, we do not distinguish the following first two closures, but we do distinguish the third closure from the first two.

$$
\begin{array}{rcl}
(P(x) \vee Q(y,z)) \cdot \{x \mapsto a, y \mapsto a\} & \equiv & P([a]) \vee Q([a], z) \\
(P(x) \vee Q(x,z)) \cdot \{x \mapsto a\} & \equiv & P([a]) \vee Q([a], z) \\
(P(x) \vee Q(a,z)) \cdot \{x \mapsto a\} & \equiv & P([a]) \vee Q(a, z)
\end{array}
$$

The $\mathcal{BS}$ calculus is parameterized with an admissible ordering on terms and a selection function for negative literals.

An ordering $\succ$ on terms is *admissible for* $\mathcal{BS}$ if $\succ$ is a *reduction ordering* that is total on ground terms with $\mathtt{tt}$ the smallest element. This ordering is extended to an ordering on literals and clauses as follows. Literals are compared by identifying $s \approx t$ with a multiset $\{\{s\}, \{t\}\}$ and $s \not\approx t$ with a multiset $\{\{s, t\}\}$, and by comparing these multisets using the twofold multiset extension of the term ordering. Clauses are compared by using the multiset extension of the ordering on literals,

Table II. Inference Rules of the $\mathcal{BS}$ Calculus

Positive Superposition: $$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \approx v) \cdot \rho}{(C \vee D \vee w[t]_p \approx v) \cdot \theta}$$	(i) $\sigma = \mathsf{MGU}(s\rho, w|_p\rho)$ 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 in $(C \vee s \approx t) \cdot \theta$; (iv) $(w \approx v) \cdot \theta$ is strictly eligible in $(D \vee w \approx v) \cdot \rho$; (v) $s\theta \approx t\theta \not\succeq w\theta \approx v\theta$; (vi) $w|_p$ is not a variable.
Negative Superposition: $$\frac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \not\approx v) \cdot \rho}{(C \vee D \vee w[t]_p \not\approx v) \cdot \theta}$$	(i) $\sigma = \mathsf{MGU}(s\rho, w|_p\rho)$ 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 in $(C \vee s \approx t) \cdot \theta$; (iv) $(w \not\approx v) \cdot \theta$ is eligible in $(D \vee w \not\approx v) \cdot \theta$; (v) $w|_p$ is not a variable.
Reflexivity Resolution: $$\frac{(C \vee s \not\approx t) \cdot \rho}{C \cdot \theta}$$	(i) $\sigma = \mathsf{MGU}(s\rho, t\rho)$ and $\theta = \rho\sigma$; (ii) $(s \not\approx t) \cdot \theta$ is eligible in $(C \vee s \not\approx t) \cdot \theta$.
Equality Factoring: $$\frac{(C \vee s \approx t \vee s' \approx t') \cdot \rho}{(C \vee t \not\approx t' \vee s' \approx t') \cdot \theta}$$	(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 in $(C \vee s \approx t \vee s' \approx t') \cdot \theta$.

and closures are compared by treating each closure $C \cdot \sigma$ as a clause $C\sigma$. We denote the orderings on literals and clauses also with $\succ$.

A *selection function* is a function that assigns to each closure $C \cdot \sigma$ a (possibly empty) submultiset of its negative literals; these literals are said to be *selected*.

A literal $L \cdot \sigma$ is *maximal* (*strictly maximal*) w.r.t. a closure $C \cdot \sigma$ if no $L' \in C$ exists such that $L'\sigma \succ L\sigma$ ($L'\sigma \succeq L\sigma$). A literal $L \cdot \sigma$ is *eligible* (*strictly eligible*) in a closure $(C \vee L) \cdot \sigma$ if either (i) $L \cdot \sigma$ is selected in $(C \vee L) \cdot \sigma$, or (ii) no literal is selected in $(C \vee L) \cdot \sigma$ and $L \cdot \sigma$ is maximal (strictly maximal) w.r.t. $C \cdot \sigma$.

The *inference rules* of $\mathcal{BS}$ are presented in Table II.[3] It is important to distinguish an *inference rule* from an *inference*. An inference rule can be understood as a template that specifies actions to be applied to any premises. An inference is an application of an inference rule to actual premises. As usual in resolution theorem proving, we assume that the common variables of the premises in each inference have been renamed apart, so the premises are variable-disjoint.

---

[3] The version of $\mathcal{BS}$ presented here can be seen as a weaker variant of the calculi from [6, 36]. Unlike [6], we ignore induced substitution positions and, unlike [36], we do not inherit ordering constraints. Weaker inference rules may result in more possible inferences; the calculus, however, remains refutationally complete.

The $\mathcal{BS}$ calculus is refutationally complete for every admissible ordering and every selection function. Moreover, the calculus is compatible with a powerful notion of redundancy, which can be used to justify many simplification rules. We assume the reader to be familiar with the basic notions from term rewriting [3]. Let $\mathcal{R}$ be a ground and convergent rewrite system and $C \cdot \sigma$ a ground closure. A variable x in the skeleton C is *variable irreducible w.r.t. $\mathcal{R}$* if (i) $x\sigma$ is irreducible by $\mathcal{R}$, or (ii) x occurs in C only in literals of the form $x \approx s$ such that $x\sigma \succ s\sigma$, and $x\sigma$ is irreducible by those rules $l \Rightarrow r \in \mathcal{R}$ for which $x\sigma \approx s\sigma \succ l \approx r$. Furthermore, $C \cdot \sigma$ is *variable irreducible w.r.t. $\mathcal{R}$* if all variables from C are variable irreducible w.r.t. $\mathcal{R}$. For $C \cdot \sigma$ a possibly nonground closure, $\mathsf{irred}_{\mathcal{R}}(C \cdot \sigma)$ is the set of all ground closures $C \cdot \sigma\tau$ that are variable irreducible w.r.t. $\mathcal{R}$. For N a set of closures, we set $\mathsf{irred}_{\mathcal{R}}(N) = \bigcup_{C \cdot \sigma \in N} \mathsf{irred}_{\mathcal{R}}(C \cdot \sigma)$.

A closure $C \cdot \sigma$ is *redundant w.r.t. a set of closures N* if, for all rewrite systems $\mathcal{R}$ and all ground substitutions τ, if $C \cdot \sigma\tau \in \mathsf{irred}_{\mathcal{R}}(C \cdot \sigma)$, then $\mathsf{irred}_{\mathcal{R}}(N)$ contains closures $C_1, \ldots, C_n$ such that $\{C_1, \ldots, C_n\} \models C\sigma\tau$ and $C\sigma\tau \succ C_i$. In [6, 36], the authors also define the notion of redundancy for inferences. This definition is rather technical and is not used in our results, so please refer to [6, 36] for details. A set of closures N is *saturated under $\mathcal{BS}$* if each $\mathcal{BS}$ inference from N is redundant w.r.t. N. It is known that, if each $\mathcal{BS}$ inference with closures from N produces a redundant conclusion, then N is saturated up to redundancy.

The notion of redundancy makes it possible to extend $\mathcal{BS}$ with *simplification rules* without losing completeness. These rules simplify a closure set $N \cup \{C \cdot \rho\}$ into one of the m closure sets $N \cup N^j$, $1 \leq j \leq m$. A simplification rule is *deterministic* if $m = 1$; it is *nondeterministic* otherwise. A simplification rule is *consistency preserving* (or *sound*) if, for every satisfiable $N \cup \{C \cdot \rho\}$, some resulting closure set $N \cup N^j$ is satisfiable as well. A simplification rule is *compatible with the redundancy criterion* (or *correct*) if, for every $N \cup \{C \cdot \rho\}$ and every j with $1 \leq j \leq m$, the closure $C \cdot \rho$ is redundant w.r.t. $N \cup N^j$.

Some of the simplification rules of $\mathcal{BS}$ rely on the notion of η-*domination*: for a substitution η, a term $s \cdot \sigma$ is η-dominated by a term $t \cdot \theta$, written $s \cdot \sigma \sqsubseteq_\eta t \cdot \theta$, if and only if (i) $s\sigma\eta = t\theta$, and (ii) a substitution ρ exists such that $s = t\rho$. For example,

$$f(g(x)) \sqsubseteq_{\{x \mapsto a\}} f(y) \cdot \{y \mapsto g(a)\} \equiv f([g(a)]).$$

For literals, $(s \approx t) \cdot \sigma \sqsubseteq_\eta (w \approx v) \cdot \theta$ if and only if $s \cdot \sigma \sqsubseteq_\eta w \cdot \theta$ and $t \cdot \sigma \sqsubseteq_\eta v \cdot \theta$, or $s \cdot \sigma \sqsubseteq_\eta v \cdot \theta$ and $t \cdot \sigma \sqsubseteq_\eta w \cdot \theta$. The definition is analogous for negative literals. For closures, $C \cdot \sigma \sqsubseteq_\eta D \cdot \theta$ if and only if an injective function λ from literals of C to literals of D exists such that $L \cdot \sigma \sqsubseteq_\eta L' \cdot \theta$ for each $L \in C$ and $L' = \lambda(L)$.

Next, we present an overview of sound and correct simplification rules known in the literature. All but the last one are deterministic.

Strict closure subsumption: $N \cup \{C \cdot \sigma\}$ is simplified into N if a closure $D \cdot \theta$ in N and a substitution η exist such that $D \cdot \theta \sqsubseteq_\eta C \cdot \sigma$ and D contains fewer literals than C. For example, the closure

$$(P(f(x)) \vee Q(y)) \cdot \{x \mapsto a, y \mapsto f(a)\} \equiv P(f([a])) \vee Q([f(a)])$$

is subsumed by the closures $P(f(x))$, $Q(f(x))$, and $Q([f(x)])$ but not by the closure $P(x)$.[4]

Elimination of duplicate literals: $N \cup \{(C \vee L_1 \vee L_2) \cdot \sigma\}$ is simplified into $N \cup \{(C \vee L_1) \cdot \sigma\}$ if $L_1 \cdot \sigma \sqsubseteq_{\{\}} L_2 \cdot \sigma$.

Syntactic tautology deletion: $N \cup \{C \cdot \sigma\}$ is simplified into N if $C \cdot \sigma$ is a *syntactic tautology*; the latter is the case if $C \cdot \sigma$ contains a literal $(s \approx t) \cdot \sigma$ such that $s\sigma = t\sigma$, or a pair of literals $(s \approx t) \cdot \sigma$ and $(s' \not\approx t') \cdot \sigma$ such that $s\sigma = s'\sigma$ and $t\sigma = t'\sigma$.

Ground unit resolution: $N \cup \{(C \vee L) \cdot \rho\}$ is simplified into a closure set $N \cup \{C \cdot \rho\}$ if N contains a ground closure $L' \cdot \theta$ such that $L'\theta = \overline{L}\rho$.

Trivial reflexivity resolution: $N \cup \{(C \vee s \not\approx t) \cdot \rho\}$ is simplified into $N \cup \{C \cdot \rho\}$ provided that $s\rho = t\rho$.

Ground unit cut: N is nondeterministically simplified into either $N \cup \{L \cdot \rho\}$ or $N \cup \{\overline{L} \cdot \rho\}$ for $L\rho$ a ground literal.[5]

The following rule is not a simplification rule; however, it is well-known not to affect soundness or completeness of $\mathcal{BS}$.

Elimination of marked position (called *retraction* in [6]): $N \cup \{C \cdot \sigma\}$ can be replaced with $N \cup \{C\rho \cdot \theta\}$ for ρ and θ such that $\sigma = \rho\theta$. For example, the closure

$$(P(x) \vee Q(x)) \cdot \{x \mapsto f(a)\} \equiv P([f(a)]) \vee Q([f(a)])$$

can be replaced with

$$(P(x) \vee Q(f(y))) \cdot \{x \mapsto f(a), y \mapsto a\} \equiv P([f(a)]) \vee Q(f([a])).$$

As usual, we use the standard ordered resolution rule as a "macro" that combines negative superposition with eager application of the trivial reflexivity resolution rule [4].

A $\mathcal{BS}$ *derivation from* a closure set N_0 is a finitely branching tree whose nodes are labeled with closure sets such that (*i*) the root of the tree is labeled by N_0 and (*ii*) each node labeled with a closure set N is either a leaf node if N is saturated by $\mathcal{BS}$ up to redundancy, or it has

---

[4] Note that the classical subsumption is not an instance of this rule; for example, $P(x)$ does not subsume $P(a) \vee P(b)$ in $\mathcal{BS}$, since $P(x) \not\sqsubseteq_\eta P(a) \vee P(b)$ for any η.

[5] Ground unit cut rule does not have a premise being simplified, so it is not really a simplification rule; we treat it as such for the ease of presentation.

m children labeled by the closure sets $N^1, \ldots, N^m$ that are obtained by applying a nonredundant inference or a simplification rule to N. Each derivation is required to be *fair*; intuitively, no inference should be postponed infinitely often. For a precise definition of fairness and for ways of achieving it, please refer to [5, 48]. $\mathcal{BS}$ is sound [6, 36]: if some derivation from N_0 contains the empty closure in the label of each leaf, then N_0 is unsatisfiable. Furthermore, $\mathcal{BS}$ is also complete [6, 36]: if N_0 is unsatisfiable, then each fair derivation from N_0 is finite and it contains the empty closure in the label of each leaf.

3. Outline of the Decision Procedure

In this section, we discuss the problems encountered in applying the techniques of resolution decision procedures to $\mathcal{SHOIQ}$. We also discuss the solutions to these problems and thus outline the main aspects of our decision procedure.

The basic principles for deciding a fragment $\mathcal{L}$ of first-order logic by (a refinement of) resolution were first formulated in [28]. The general principle is to check satisfiability of a formula $\varphi \in \mathcal{L}$ using a calculus $\mathcal{C}$ that is sound and complete for first-order clausal logic. A formula φ is translated into an equisatisfiable set of clauses, which are then saturated by $\mathcal{C}$. Soundness and completeness of $\mathcal{C}$ guarantee that φ is unsatisfiable if and only if the empty clause is derived in the saturation. Hence, satisfiability of φ can be decided by $\mathcal{C}$ provided that saturation always terminates.

To prove termination of the saturation procedure, one typically identifies a class of clauses $\mathcal{N}_{\mathcal{L}}$ such that (i) for a finite signature, $\mathcal{N}_{\mathcal{L}}$ contains finitely many clauses different up to variable renaming, (ii) every formula $\varphi \in \mathcal{L}$ can be translated into clauses from $\mathcal{N}_{\mathcal{L}}$ in a satisfiability-preserving manner, and (iii) $\mathcal{N}_{\mathcal{L}}$ is closed under $\mathcal{C}$—that is, every inference of $\mathcal{C}$ with premises from $\mathcal{N}_{\mathcal{L}}$ produces only clauses from $\mathcal{N}_{\mathcal{L}}$. Provided that the inferences do not extend the signature, these conditions guarantee that every saturation produces at most finitely many clauses and thus necessarily terminates.

Calculi such as ordered resolution [5] or superposition calculi [37] have parameters (e.g., the selection function or the term ordering) that must be specified before performing any inferences. These parameters determine which inferences are performed with clauses from $\mathcal{N}_{\mathcal{L}}$. Hence, the proper selection of the parameters—that is, defining a suitable *saturation strategy*—plays the key role in achieving (iii).

These principles have been used to obtain decision procedures for various well-known first-order fragments, such as the guarded fragment

[18, 14], the two-variable fragment [15], the monadic fragment [7], clause classes like $\mathbf{E}^+$ [45, 12, 13], $\mathcal{PVD}$ [32], $\mathcal{PVD}^g_=$ [41], Maslov class $\mathbf{K}$ [27], various description and modal logics [38], and various first-order theories [1]. An overview of some of these results is given in [16, 17].

To decide satisfiability of a $\mathcal{SHOIQ}$ knowledge base KB, we use the basic superposition calculus described in Section 2.3. Our procedure consists of two phases. In the *preprocessing phase*, KB is translated into an equisatisfiable set of closures $\Gamma(KB)$, and in the *saturation phase*, the inference rules of $\mathcal{BS}$ are applied to $\Gamma(KB)$ according to a particular strategy. In the rest of this section, we discuss the essential steps of these phases.

3.1. Preprocessing Phase

The preprocessing phase of our procedure consists of three steps: (i) the elimination of transitivity axioms from KB, (ii) the translation of axioms into closures, and (iii) the introduction of guards.

3.1.1. *Elimination of Transitivity Axioms*
Transitivity axioms are known to be difficult to deal with in saturation-based theorem proving. Consider the following $\mathcal{SHOIQ}$ knowledge base and its translation into closures.

$$(3) \qquad \top \sqsubseteq \exists R.\top \qquad R(x, f(x))$$
$$(4) \qquad \mathsf{Trans}(R) \qquad \neg R(x, y) \vee \neg R(y, z) \vee R(x, z)$$

One of the negative literals in (4) should be eligible because both of these literals are maximal: either one of them is selected, or both of them are eligible. If the first literal is eligible, then ordered resolution derives closures of unbounded depth, as shown in (5)–(7). In the following examples, we underline the part of the literal that is unified in inferences. Furthermore, $R[xx; yy]$ ($S[xx; yy]$) next to a closure means that the closure is derived by resolution (superposition) of closures labeled with (xx) and (yy).

$$(5) \qquad \underline{\neg R(x, y)} \vee \neg R(y, z) \vee R(x, z)$$
$$(6) \qquad \underline{\neg R(f(x), z)} \vee R(x, z) \quad \text{R}[3;5]$$
$$(7) \qquad \underline{R(x, f(f(x)))} \quad \text{R}[3;6]$$
$$(8) \qquad \underline{\neg R(f(f(x)), z)} \vee R(x, z) \quad \text{R}[7;5] \; etc.$$

If the second literal in (4) is eligible, then ordered resolution derives closures with an unbounded number of variables, as shown in (9)–(11).

$$(9) \qquad \neg R(x, y) \vee \underline{\neg R(y, z)} \vee R(x, z)$$

$$(10) \qquad\qquad \neg R(x, y) \lor \underline{R(x, f(y))} \quad \text{R[3;9]}$$

$$(11) \qquad \neg R(x, y) \lor \neg R(y, y_1) \lor \underline{R(x, f(y_1))} \quad \text{R[10;9]} \; etc.$$

To avoid such problems, simplification rules were developed in [30] that decompose certain conclusions of inferences with clauses of the form (5). In the case of description logics, however, there is a simpler solution: the knowledge base KB can be polynomially transformed into an equisatisfiable $\mathcal{ALCHOIQ}$ knowledge base $\Omega(KB)$. We present this transformation in detail in Section 4.1.

3.1.2. *Translation of Axioms into Closures*

A naïve translation of $\Omega(KB)$ into closures can incur an exponential blowup in the number of closures. Consider the following axiom:

$$(12) \qquad A \sqsubseteq (A_1 \sqcap B_1) \sqcup (A_2 \sqcap B_2) \sqcup \ldots \sqcup (A_n \sqcap B_n).$$

Direct translation of (12) into conjunctive normal form distributes the disjunctions over the conjunctions in the right-hand side, thus producing 2^n conjuncts. This can be avoided by using the *structural transformation*, also known as *renaming* [42]. This transformation transforms (12) into the following axioms, where $Q_1, \ldots, Q_n$ are fresh concept names.

$$(13) \qquad\qquad A \sqsubseteq Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_n$$

$$(14) \qquad\qquad Q_i \sqsubseteq A_i \sqcap B_i \quad 1 \le i \le n$$

The axioms (13)–(14) can be straightforwardly translated into conjunctive normal form without an exponential blowup. We present the structural transformation in detail in Section 4.2.

The application of the operator π from Table I to the result of the structural transformation produces first-order formulae with counting quantifiers, which can be expressed by using ordinary quantifiers and equality according to (1) and (2). When translating such formulas into closures, the variables $x_1, \ldots, x_n$ in (1) are Skolemized; however, the variables $x_1, \ldots, x_{n+1}$ in (2) become a part of the resulting closure. Although it might be possible to handle such closures by using the hyperresolution inference rule as it was done for the logic $\mathcal{SHIQ}^-$ in [25], in this paper we use the following alternative first-order translation of $\exists^{\le n} x.\varphi(x, \overline{y})$.

$$(15) \qquad \exists^{\le n} x.\varphi(x, \overline{y}) \;=\; \exists x_1, \ldots, x_n.\forall x. \left[\varphi(x, \overline{y}) \to \bigvee_{i=1}^{n} x \approx x_i \right]$$

One can easily see that (15) is equivalent to (2). The advantage of (15) over (2) is that, instead of $n + 1$ universally quantified variables

$x_1, \ldots, x_{n+1}$, we now have only one universally quantified variable x, which reduces the number of variables in the resulting closures.

3.1.3. *Introduction of Guards*

Nominals can be used to restrict the cardinality of the interpretation domain. For example, the following axiom ensures that the interpretation domain contains at most two elements.

$$(16) \qquad\qquad \top \sqsubseteq \{a_1\} \sqcup \{a_2\}$$

This axiom is translated into the following closure:

$$(17) \qquad\qquad x \approx \underline{a_1} \vee x \approx a_2.$$

In (17), the variable x is *unshielded*—that is, it does not occur as a proper subterm of some term. This allows for superposition inferences from a_i, since $x \succ a_i$ does not hold for any admissible ordering $\succ$. Inferences with (17) can produce closures with an arbitrary number of variables (w.l.o.g. we can assume that $a_1 \succ a_2$):

$$(18) \qquad\qquad x_1 \approx \underline{a_2} \vee x_2 \approx a_2 \vee x_1 \approx x_2 \quad \text{S}[17;17]$$

$$(19) \quad x_2 \approx \underline{a_2} \vee x_3 \approx a_2 \vee x_1 \approx x_2 \vee x_1 \approx x_3 \vee x_3 \approx x_4 \quad \text{S}[18;18] \; etc.$$

We avoid such problems by using another satisfiability-preserving transformation, which we call the *introduction of guards*. Intuitively, we introduce a new unary predicate symbol T and add closures that make T hold on all elements of the Herbrand universe. Then, we add a literal $\neg T(x)$ to every closure where x occurs unshielded. In our example, the closure (17) is replaced with the following closure.

$$(20) \qquad\qquad \neg \underline{T(x)} \vee x \approx a_1 \vee x \approx a_2$$

Now the variable x in (20) is shielded, so the inference on a_1 can be avoided by selecting the guard literal $\neg T(x)$. A similar strategy for transforming clauses into *range-restricted* ones (i.e., the clauses in which all variables are guarded) has been presented in [8]. We present the details of our transformation in Section 4.3.

3.2. Saturation Phase

As we discuss in this section, a straightforward saturation of the closures obtained by preprocessing by $\mathcal{BS}$ does not necessarily terminate. Therefore, we extend $\mathcal{BS}$ with four new simplification rules that deal with dangerous closures. We first discuss the problems due to an interaction between number restrictions, role hierarchy, and inverse roles.

14

We then consider the problems due to an interaction between nominals, inverse roles, and number restrictions.

3.2.1. *Number Restrictions, Role Hierarchy, and Inverse Roles*

Even without nominals, the combination of number restrictions, role hierarchy, and inverse roles is difficult to handle using resolution. Consider the following knowledge base KB and its translation into closures.

(21) $\quad \top \sqsubseteq \exists S_i.\top \ \ [i = 1, 2, 3] \quad \underline{S_i(x, f_i(x))}$

(22) $\quad \top \sqsubseteq \ \leqslant 1\, S_i.\top \ \ [i = 1, 2, 3] \quad \neg \underline{S_i(x, y)} \vee y \approx s_i(x)$

(23) $\quad \quad S_i^- \sqsubseteq S_i \ \ [i = 1, 2, 3] \quad \neg \underline{S_i^-(x, y)} \vee S_i(x, y)$

(24) $\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \neg \underline{S_i(x, y)} \vee S_i^-(y, x)$

(25) $\quad \quad \quad S_i \sqsubseteq R \ \ [i = 1, 2, 3] \quad \neg \underline{S_i(x, y)} \vee R(x, y)$

(26) $\quad \top \sqsubseteq \ \leqslant 2\, R.\top \quad \quad \quad \quad \neg \underline{R(x, y)} \vee y \approx h_1(x) \vee y \approx h_2(x)$

By (22), the roles S_1, S_2, and S_3 are functional; by (23), they are symmetric; and by (25), they are subroles of the role R. Assuming that the term ordering is such that $s_i(x) \succ f_i(x)$, we can derive the following closures.

(27) $\quad \quad \quad \quad \quad [f_i(x)] \approx \underline{s_i(x)} \ \ [i = 1, 2, 3] \ \ \text{R}[21;22]$

(28) $\quad \quad \quad \quad \quad \underline{S_i^-([f_i(x)], x)} \ \ [i = 1, 2, 3] \ \ \text{R}[21;24]$

(29) $\quad \quad \quad \quad \quad \quad \underline{S_i([f_i(x)], x)} \ \ [i = 1, 2, 3] \ \ \text{R}[28;23]$

(30) $\quad \quad \quad \quad \quad \underline{s_i([f_i(x)])} \approx x \ \ [i = 1, 2, 3] \ \ \text{R}[29;22]$

(31) $\quad \quad \quad \quad \quad \underline{f_i([f_i(x)])} \approx x \ \ [i = 1, 2, 3] \ \ \text{S}[30;27]$

Note that, if we had $s_i(x) \not\succ f_i(x)$ for some i, then we would derive $S_i(x, s_i(x))$ and, consequently, $s_i(s_i(x)) \approx x$. This closure is analogous to (21) and (31), but with f_i replaced by s_i, so the rest of this example would follow analogously; that is, adjusting the term ordering does not correct our problem.

The inferences are continued as follows, assuming that $h_j(x) \succ f_i(x)$ for $1 \le i \le 3$ and $1 \le j \le 2$.

(32) $\quad \quad \quad \quad \quad \quad \underline{R(x, [f_i(x)])} \ \ [i = 1, 2, 3] \ \ \text{R}[21;25]$

(33) $\quad \quad f_i(x) \approx \underline{h_1(x)} \vee f_i(x) \approx h_2(x) \ \ [i = 1, 2, 3] \ \ \text{R}[32;26]$

(34) $\quad \underline{f_1(x)} \approx f_2(x) \vee f_1(x) \approx f_3(x) \vee f_2(x) \approx f_3(x) \ \ \text{S}[33;33]$

The closure (34) is derived by superposition between (33) for different i and by elimination of duplicate literals. We further derive the following

closures, in which we enclose "interesting" literals in frames.

$$(35) \qquad f_2([f_1(x)]) \approx x \vee \boxed{f_1([f_1(x)]) \approx f_3([f_1(x)])} \vee \ \text{S[31;34]}$$
$$\overline{f_2([f_1(x)]) \approx f_3([f_1(x)])}$$

$$(36) \qquad f_2([f_1(x)]) \approx x \vee \boxed{S_i([f_1(x)], f_3([f_1(x)]))} \vee \ \text{S[35;21]}$$
$$f_2([f_1(x)]) \approx \overline{f_3([f_1(x)])} \qquad [i = 1, 2, 3]$$

One can superpose the deep closure (31) into (34), resulting in a closure (35) with an equation between terms of depth two. Such a literal is dangerous for superposition, however, since it can easily result in deeper closures: by superposing (35) into (21), we obtain a closure (36), which has a literal $S_i([f_1(x)], f_3([f_1(x)]))$ of depth three. Such a literal can participate in similar inferences as (21) and can produce even deeper closures. Note that superposition from (31) into (21) does not result in a deeper closure, although (31) is of depth two as well. Also, choosing a different ordering (e.g., with $f_i(x) \succ h_1(x)$ for some i) does not solve the problem, since a closure similar to (35) would be derived by superposition from (31) into (33). Note that the basic restriction, which blocks superposition into marked positions, does not help us resolve this problem.

A similar problem was studied in [29] for the guarded fragment with counting quantifiers. The solution was to introduce a new simplification rule called *splitting through new propositional symbols*. Instead of waiting for (36) to be derived, (34) is eagerly split into these closures:

$$(37) \qquad Q_{f_1,f_2}(x) \vee Q_{f_2,f_3}(x) \vee Q_{f_1,f_3}(x)$$
$$(38) \qquad \neg Q_{f_i,f_j}(x) \vee f_i(x) \approx f_j(x) \qquad [1 \leq i < j \leq 3]$$

The predicate symbols $Q_{f_i,f_j}(x)$ are fresh and unique for a pair of f_i and f_j. Since the number of different such pairs is finite, the number of these new predicate symbols is bounded. Furthermore, superposition inferences from (31) into (38) do not produce equational literals such as the ones in (35). In [26], a similar problem was solved for the DL $\mathcal{SHIQ}$ by introducing a *decomposition rule*, which is similar to the splitting rule. In Section 5.1, we adapt these results and extend basic superposition with Decomposition 1 and 2 simplification rules.

3.2.2. *Nominals, Number Restrictions, and Inverse Roles*

The extension of $\mathcal{SHIQ}$ to $\mathcal{SHOIQ}$ introduces a new problem for resolution due to an interaction of nominals, number restrictions, and inverse roles [23]. Consider the following knowledge base and its translation into closures.

$$(39) \qquad\qquad O \sqsubseteq \{c\} \qquad\qquad \neg \underline{O(x)} \vee x \approx c$$

$$(40) \qquad A \sqsubseteq B_1 \sqcup B_2 \qquad\qquad \neg \underline{A(x)} \vee B_1(x) \vee B_2(x)$$

$$(41) \qquad B_i \sqsubseteq \exists R_i.A \quad [i = 1, 2] \quad \neg \underline{B_i(x)} \vee \underline{R_i(x, f_i(x))}$$

$$(42) \qquad\qquad\qquad\qquad\qquad \neg B_i(x) \vee \underline{A(f_i(x))}$$

$$(43) \qquad \top \sqsubseteq \, \leqslant 1\, R_i^-.\top \quad [i = 1, 2] \quad \neg \underline{R_i^-(x, y)} \vee g_i(x) \approx y$$

$$(44) \qquad O \sqsubseteq \forall R_i.O \quad [i = 1, 2] \quad \neg O(x) \vee \neg \underline{R_i(x, y)} \vee O(y)$$

$$(45) \qquad\qquad\qquad [i = 1, 2] \quad \neg \underline{R_i(x, y)} \vee R_i^-(y, x)$$

Saturation of these closures by $\mathcal{BS}$ produces the following closures.

$$(46) \qquad\qquad \neg B_i(x) \vee \underline{B_1([f_i(x)])} \vee B_2([f_i(x)] \quad \text{R[40;42]}$$

$$(47) \qquad\qquad \neg B_i(x) \vee \neg O(x) \vee \underline{O([f_i(x)])} \quad \text{R[41;44]}$$

$$(48) \qquad\qquad \boxed{\neg B_i(x) \vee \neg O(x) \vee \underline{[f_i(x)]} \approx c} \quad \text{R[47;39]}$$

$$(49) \qquad\qquad \neg B_i(x) \vee \neg O(x) \vee \underline{R_i(x, c)} \quad \text{S[48;41]}$$

$$(50) \qquad\qquad \neg B_i(x) \vee \neg O(x) \vee \underline{R_i^-(c, x)} \quad \text{R[49;45]}$$

$$(51) \qquad\qquad \neg \underline{B_i(x)} \vee \neg O(x) \vee x \approx g_i(c) \quad \text{R[50;43]}$$

$$(52) \qquad \neg B_i(x) \vee \underline{B_2([f_i(x)])} \vee \neg O([f_i(x)]) \vee [f_i(x)] \approx g_1(c) \quad \text{R[46;51]}$$

$$(53)\ \neg B_i(x) \vee \neg \underline{O([f_i(x)])} \vee [f_i(x)] \approx g_1(c) \vee [f_i(x)] \approx g_2(c) \quad \text{R[52;51]}$$

$$(54) \qquad \boxed{\neg B_i(x) \vee \neg O(x) \vee \underline{[f_i(x)]} \approx g_1(c) \vee [f_i(x)] \approx g_2(c)} \quad \text{R[47;53]}$$

The closure (54) is similar in structure to (48): it contains just two literals, $f_i(x) \approx g_1(c)$ and $f_i(x) \approx g_2(c)$, instead of one literal, $f_i(x) \approx c$. One can easily see that all inferences with (48) can be repeated for (54) and that this would produce even longer closures with even deeper literals $f_i(x) \approx g_1(g_1(c))$, $f_i(x) \approx g_2(g_1(c))$ and so on. This clearly prevents the saturation from terminating.

To deal with this problem, we simplify (54) into

$$(55) \qquad\qquad \neg B_i(x) \vee \neg O(x) \vee c \approx g_1(c) \vee c \approx g_2(c).$$

The closure (55) is a logical consequence of (48) and (54). Furthermore, (55) makes (54) redundant, since (54) follows from the smaller closures (48) and (55). Thus, (54) can be deleted from the closure set, which eventually ensures termination of the saturation. In Section 5.1, we generalize this idea and introduce two simplification rules, called Nominal Generation 1 and 2.

4. Computing the Set of Closures

We now present the preprocessing phase of our algorithm in detail. The main task of this phase is to translate a $\mathcal{SHOIQ}$ knowledge base KB into a suitable set of closures $\Gamma(KB)$.

4.1. Eliminating Transitivity Axioms

As explained in Section 3.1.1, we eliminate transitivity axioms from KB by polynomially encoding KB into an equisatisfiable $\mathcal{ALCHOIQ}$ knowledge base $\Omega(KB)$.

DEFINITION 1. *For KB a $\mathcal{SHOIQ}$ knowledge base, $\mathsf{clos}(KB)$ is the smallest set of concepts such that*

- $\mathsf{nnf}(\neg C \sqcup D) \in \mathsf{clos}(KB)$ *if $C \sqsubseteq D \in KB_{\mathcal{T}}$;*

- $\mathsf{nnf}(C) \in \mathsf{clos}(KB)$ *if $C(a) \in KB_{\mathcal{A}}$;*

- $D \in \mathsf{clos}(KB)$ *if $C \in \mathsf{clos}(KB)$ and D occurs in C;*

- $\mathsf{nnf}(\neg C) \in \mathsf{clos}(KB)$ *if $\leqslant n\,R.C \in \mathsf{clos}(KB)$;*

- $\forall S.C \in \mathsf{clos}(KB)$ *if $\forall R.C \in \mathsf{clos}(KB)$ and S is such that $S \sqsubseteq^* R$ and $\mathsf{Trans}(S) \in KB_{\mathcal{R}}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in KB_{\mathcal{R}}$.*

*The $\mathcal{ALCHOIQ}$ knowledge $\Omega(KB)$ is obtained from KB by (*i*) removing all transitivity axioms and (*ii*) adding an axiom $\forall R.C \sqsubseteq \forall S.(\forall S.C)$ for each concept $\forall R.C \in \mathsf{clos}(KB)$ and each role S with $S \sqsubseteq^* R$ and $\mathsf{Trans}(S) \in KB_{\mathcal{R}}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in KB_{\mathcal{R}}$.*

This encoding is polynomial in $|KB|$: the number of concepts in $\mathsf{clos}(KB)$ stemming from a concept C is bounded by $2 \cdot |C| \cdot |N_R|$ and, for each concept from $\mathsf{clos}(KB)$, we generate at most $|N_R|$ axioms in the TBox of $\Omega(KB)$.

This encoding is similar to the transformation of formulas of modal logic K4 into formulas of modal logic K from [44]. Another related algorithm for transforming $\mathcal{SHIQ}$ concepts into $\mathcal{ALCIQ}b$ concepts was presented in [46]. For the proof of correctness of this transformation, we refer the interested reader to [34].[6]

LEMMA 2 ([34]). *KB is satisfiable if and only if $\Omega(KB)$ is satisfiable.*

---

[6] The proof is given for $\mathcal{SHIQ}$, but the extension to $\mathcal{SHOIQ}$ is trivial.

Table III. The Structural Transformation of KB

$$\Theta(KB) = \bigcup_{\alpha \in KB'_{\mathcal{R}} \cup KB'_{\mathcal{A}}} \Theta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in KB'_{\mathcal{T}}} \Theta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2))$$

$$\Theta(a \approx b) = \{a \approx b\}$$

$$\Theta(a \not\approx b) = \{a \not\approx b\}$$

$$\Theta(C(a)) = \{Q_C(a)\} \cup \Theta(Q_C \sqsubseteq \mathsf{nnf}(C))$$

$$\Theta(R(a,b)) = \{R(a,b)\}$$

$$\Theta(R \sqsubseteq S) = \{R \sqsubseteq S\}$$

$$\Theta(A \sqsubseteq B) = \{A \sqsubseteq B\}$$

$$\Theta(A \sqsubseteq \neg B) = \{A \sqsubseteq \neg B\}$$

$$\Theta(A \sqsubseteq \{a\}) = \{A \sqsubseteq \{a\}\}$$

$$\Theta(A \sqsubseteq \neg\{a\}) = \{\neg A(a)\}$$

$$\Theta(A \sqsubseteq C_1 \sqcap C_2) = \Theta(A \sqsubseteq C_1) \cup \Theta(A \sqsubseteq C_2)$$

$$\Theta(A \sqsubseteq C_1 \sqcup C_2) = \{A \sqsubseteq Q_{C_1} \sqcup Q_{C_2}\} \cup \Theta(Q_{C_1} \sqsubseteq C_1) \cup \Theta(Q_{C_2} \sqsubseteq C_2)$$

$$\Theta(A \sqsubseteq \exists R.C) = \{A \sqsubseteq \exists R.Q_C\} \cup \Theta(Q_C \sqsubseteq C)$$

$$\Theta(A \sqsubseteq \forall R.C) = \{A \sqsubseteq \forall R.Q_C\} \cup \Theta(Q_C \sqsubseteq C)$$

$$\Theta(A \sqsubseteq \geqslant n\,R.C) = \{A \sqsubseteq \geqslant n\,R.Q_C\} \cup \Theta(Q_C \sqsubseteq C)$$

$$\Theta(A \sqsubseteq \leqslant n\,R.C) = \{A \sqsubseteq \leqslant n\,R.\neg Q_D\} \cup \Theta(Q_D \sqsubseteq D) \text{ for } D = \mathsf{nnf}(\neg C)$$

Note: $KB' = \Omega(KB)$; A and B are concept names or $\top$; C, C_1, and C_2 are arbitrary concepts; R and S are roles; and Q_X is a new concept name not occurring in KB that is unique for the concept X.

4.2. Translation into Closures

Because of the reasons explained in Section 3.1, we next apply the structural transformation [42] to $\Omega(KB)$:

DEFINITION 3. *For a $\mathcal{SHOIQ}$ knowledge base KB, the result $\Theta(KB)$ of applying the* structural transformation *to KB is defined in Table III.*

The following lemma follows trivially from the fact that the transformation from Table III corresponds to the well-known structural transformation of first-order formulas.

LEMMA 4. *A $\mathcal{SHOIQ}$ knowledge base KB is satisfiable if and only if $\Theta(KB)$ is satisfiable; furthermore, $\Theta(KB)$ can be computed in time polynomial in $|KB|$.*

The axioms in $\Theta(KB)$ contain at most one concept that is not a concept name, so they can be straightforwardly converted into closures by translating them into first-order logic using the operator π from Table I, skolemizing the existential quantifiers, and translating the

Table IV. Closure Types after Preprocessing

Axiom	Closure	
$R = \mathsf{Inv}(S)$	$\neg R(x,y) \vee S(y,x)$	
	$\neg S(x,y) \vee R(y,x)$	
$R \sqsubseteq S$	$\neg R(x,y) \vee S(x,y)$	
$A \sqsubseteq \exists R.B$	$\neg A(x) \vee R(x, f(x))$	
	$\neg A(x) \vee B(f(x))$	
$A \sqsubseteq \geqslant n\, R.B$	$\neg A(x) \vee R(x, f_i(x))$	$1 \leq i \leq n$
	$\neg A(x) \vee B(f_i(x))$	$1 \leq i \leq n$
	$\neg A(x) \vee f_i(x) \not\approx f_j(x)$	$1 \leq i < j \leq n$
$A \sqsubseteq \bigsqcup(\neg)B_i$	$\neg A(x) \vee \bigvee(\neg)B_i(x)$	
$A \sqsubseteq \{c\}$	$\neg A(x) \vee x \approx c$	
$A \sqsubseteq \forall R.B$	$\neg A(x) \vee \neg R(x,y) \vee B(y)$	
$A \sqsubseteq \leqslant n\, R.\neg B$	$\neg A(x) \vee \neg R(x,y) \vee B(y) \vee \bigvee_{i=1}^{n} f_i(x) \approx y$	
$A(c)$	$A(c)$	
$R(c,d)$	$R(c,d)$	
$c \approx d$	$c \approx d$	
$c \not\approx d$	$c \not\approx d$	

Note: The function symbols $f_{(i)}$ are fresh for each axiom.

result into the conjunctive normal form. We denote the resulting set of closures with $\Xi(KB)$. Table IV shows the closures that are produced by different types of axioms.

In [34], several optimizations of the clausification algorithm were presented that can be used to reduce the size and the number of closures in $\Xi(KB)$; however, these are not essential for the correctness of our algorithm so we do not discuss them further.

4.3. INTRODUCTION OF GUARDS

The final step in preprocessing is the introduction of guards.

DEFINITION 5. *For a closure $C \cdot \rho$, a variable x from $C\rho$ is* guarded *if it occurs in $C\rho$ in a negative nonequational literal; this literal is called a* guard *for x.*

*Let KB be a $\mathcal{SHOIQ}$ knowledge base and T a fresh predicate symbol. Then, $\Gamma(KB)$ is the smallest set such that (*i*) for each closure $C \cdot \rho \in \Xi(KB)$, $\Gamma(KB)$ contains $\neg T(x_1) \vee \cdots \vee \neg T(x_n) \vee C \cdot \rho$, where $x_1, \ldots, x_n$ are all nonguarded variables of $C\rho$; (*ii*) for each constant*

c occurring in $\Xi(KB)$, $\Gamma(KB)$ *contains the closure* $T(c)$ *(if there are no constants, we add one); and (*iii*) for each unary function symbol* f *occurring in* $\Xi(KB)$, $\Gamma(KB)$ *contains the closure* $\neg T(x) \vee T(f(x))$.

LEMMA 6. $\Xi(KB)$ *is satisfiable if and only if* $\Gamma(KB)$ *is satisfiable.*

Proof. ($\Rightarrow$) Let I be a model of $\Xi(KB)$, and let I' be obtained from I by interpreting $T(x)$ to be true for all x. Clearly, I' is a model of $\Gamma(KB)$. ($\Leftarrow$) In each Herbrand model I of $\Gamma(KB)$, $\neg T(x) \vee T(f(x))$ and $T(c)$ ensure that T holds on all elements of I. Hence, each $\neg T(x_i)$ in a closure from $\Gamma(KB)$ is false in I, so I is a model of $\Xi(KB)$. $\qquad\square$

5. Saturating Closures by Basic Superposition

After preprocessing, our algorithm continues by saturating the set of closures $\Gamma(KB)$ by basic superposition. In this section we present the appropriate saturation strategy and prove that it is sound, complete, and terminating. We also estimate the complexity of our algorithm.

5.1. THE SATURATION STRATEGY

We say that N is a set of *DL-closures* if every closure in N is of some form from Table V.

LEMMA 7. *For every* $\mathcal{SHOIQ}$ *knowledge base KB,* $\Gamma(KB)$ *is a set of DL-closures.*

Proof. The set $\Xi(KB)$ contains only closures from Table IV, and, by Definition 5, each closure in $\Gamma(KB)$ contains a guard literal for each variable. Condition ($*$) holds vacuously. $\qquad\square$

Next, to obtain a procedure for checking satisfiability of $\Gamma(KB)$, we choose the appropriate parameters for $\mathcal{BS}$ and extend it with certain simplification rules. These rules can extend the signature with new predicate symbols and constants. In order to ensure that only finitely many new symbols are introduced into the signature, our rules reuse previously introduced symbols whenever possible. Thus, an application of a simplification rule depends not only on the current closure set but also on the inferences applied previously.

DEFINITION 8. *With* $\mathcal{BS}_{DL}$ *we denote the* $\mathcal{BS}$ *calculus parameterized by any admissible term ordering* $\succ$ *such that* $f(x) \succ A(x) \succ B(x) \succ c$, $R(x,c) \succ A(x)$, $R(c,x) \succ A(x)$, *and* $B(f(x)) \succ g(c)$, *for a binary predicate symbol* $R \in \mathcal{A}$ *and unary predicate symbols* $A \in \mathcal{A}$ *and*

Table V. Types of DL-Closures

1	$\alpha(x) \vee (\neg)f(x) \approx g(x)$
2	$\alpha(x) \vee (\neg)f([g(x)]) \approx x$
3	$\alpha(x) \vee (\neg)A(f(x))$
4	$\beta(x) \vee (\neg)f(x) \approx c$
5	$\beta(x) \vee \bigvee(\neg)x \approx t_i$
6	$\alpha(x) \vee \beta([f(x)]) \vee \bigvee[f(x)] \approx t_i \vee \bigvee(\neg)x \approx c_i$

Condition (∗): the disjunction $\beta([f(x)]) \vee \bigvee[f(x)] \approx t_i$ is nonempty and
        if the closure contains a literal $x \approx c_i$, then the closure set
        contains $\alpha'(x) \vee g(f(x)) \approx x$ such that $\alpha(x) = \alpha'(x) \vee \alpha''(x)$.

7	$\alpha_1(x) \vee \neg R(x,y) \vee \alpha_2(y) \vee \bigvee_{i=1}^{n} f_i(x) \approx y$	
8	$\neg R(x,y) \vee S(x,y)$ or	$\neg R(x,y) \vee S(y,x)$
9	$\alpha(x) \vee R(x,f(x))$ or	$\alpha(x) \vee R(f(x),x)$
10	$\beta(x) \vee R(x,c)$ or	$\beta(x) \vee R(c,x)$

11	unit closures and	$(\neg)B(t)$	$(\neg)t_1 \approx t_2$	$(\neg)f(g(c)) \approx d$
	the empty closure:	$(\neg)R(c,d)$	$(\neg)R(c,f(d))$	$(\neg)R(f(c),d)$ □

All predicate symbols are drawn from two sets $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{B}$ and $\mathcal{A}$ contains all predicate symbols of $\Gamma(KB)$. Each variable in each closure is guarded (see Definition 5). $\alpha(x)$ is a disjunction $(\neg)A_1(x) \vee \cdots \vee (\neg)A_n(x)$ with $A_i \in \mathcal{A}$. $\beta(x)$ is a disjunction $(\neg)B_1(x) \vee \cdots \vee (\neg)B_n(x)$ with $B_i \in \mathcal{B}$. Disjunctions $\alpha(x)$, $\beta(x)$, $\beta([f(x)])$, $\bigvee(\neg)x \approx t_i$, and $\bigvee[f(x)] \approx t_i$ may be empty. Also, c and d are constants, and $t_{(i)}$ are ground terms of the form c or $f(c)$.

$B \in \mathcal{B} \setminus \mathcal{A}$. *The selection function of* $\mathcal{BS}_{DL}$ *selects in* $C \cdot \sigma$ *a literal of the form* $\neg R(x,y)$, $x \not\approx c$, *or* $x \not\approx f(c)$; *if there are no such literals and* $C\sigma$ *does not contain a term* $f(x)$, *an atom* $R(x,c)$, *or an atom* $R(c,x)$, *it selects a literal* $\neg B(x)$ *if there is one.*

Apart from the standard $\mathcal{BS}$ *inferences,* $\mathcal{BS}_{DL}$ *eagerly applies the simplification rules from Table VI, elimination of duplicate literals, tautology deletion, closure subsumption, and ground unit resolution. Immediately after deriving a closure* $C \cdot \rho \vee L \cdot \rho$ *where* $L\rho$ *is ground,* $\mathcal{BS}_{DL}$ *applies the cut rule for* $L\rho$. *Marked positions are removed eagerly if doing so enables applying a simplification rule.*

An example of an ordering suitable for $\mathcal{BS}_{DL}$ is a Knuth-Bendix ordering [3] such that

$$\mathsf{weight}(f) > \mathsf{weight}(R) > \mathsf{weight}(A) >$$
$$\mathsf{weight}(B) > \mathsf{weight}(c) > \mathsf{weight}(\mathsf{tt})$$

for each nonconstant function symbol f, binary predicate symbol R, unary predicate symbols $A \in \mathcal{A}$ and $B \in \mathcal{B} \setminus \mathcal{A}$, and a constant $c \neq \mathsf{tt}$.

Table VI. Simplification Rules of $\mathcal{BS}_{DL}$

Decomposition 1: $$\frac{D \cdot \rho \vee L \cdot \rho}{D \cdot \rho \vee A(x), \\ \neg A(x) \vee L \cdot \rho}$$	(i) $L \cdot \rho$ is $(\neg)f(x) \approx g(x)$, $(\neg)f([g(x)]) \approx x$, $R(x, f(x))$, or $R(f(x), x)$; (ii) $D\rho$ contains a term $h(x)$; (iii) If Decomposition 1 has already been applied in the saturation to a premise with the same $L \cdot \rho$, then A is the same as in the previous application; otherwise, $A \in \mathcal{A}$ is fresh.
Decomposition 2: $$\frac{D \cdot \rho \vee f(x) \approx c}{D \cdot \rho \vee B(x), \\ \neg B(x) \vee f(x) \approx c}$$	(i) $D\rho$ contains either a term $h(x)$, or a literal $(\neg)A(x)$ with $A \in \mathcal{A}$; (ii) If Decomposition 2 has already been applied in the saturation to a premise with the same $f(x) \approx c$, then B is the same as in the previous application; otherwise, $B \in \mathcal{B} \setminus \mathcal{A}$ is fresh.
Nominal Generation 1: $$\frac{\alpha(x) \vee \bigvee_{i=1}^n [f(x)] \approx t_i}{\alpha(x) \vee \bigvee_{i=1}^k f(x) \approx c_i, \\ \alpha(x) \vee \bigvee_{j=1}^n c_i \approx t_j \\ (1 \leq i \leq k)}$$	(i) Some t_i is of the form $h(c)$; (ii) If Nominal Generation 1 has already been applied in the saturation to some closure $\alpha(x) \vee \bigvee_{i=1}^{n_1} [f(x)] \approx t_i'$ (for the same $\alpha(x)$ and f), then k and c_i are the same as in this previous application; otherwise, $k = n$ and c_i are fresh.
Nominal Generation 2: $$\frac{\alpha(x) \vee \bigvee_{i=1}^n [f(x)] \approx t_i \vee \bigvee_{i=1}^m x \approx c_i}{\alpha(x) \vee \bigvee_{i=1}^k B_i(x), \\ \neg B_i(x) \vee f(x) \approx e_i, \\ \neg B_i(x) \vee x \approx d_i, \\ \alpha(x) \vee \bigvee_{j=1}^n e_i \approx t_j \vee \bigvee_{j=1}^m d_i \approx c_j \\ (1 \leq i \leq k)}$$	(i) Some t_i is of the form $h(c)$; (ii) A closure $\alpha'(x) \vee g(f(x)) \approx x$, such that $\alpha(x) = \alpha'(x) \vee \alpha''(x)$, has been derived before; (iii) If Nominal Generation 2 has already been applied in the saturation to some closure $\alpha(x) \vee \bigvee_{i=1}^{n_1} [f(x)] \approx t_i' \vee \bigvee_{i=1}^{m_1} x \approx c_i'$ (for the same $\alpha(x)$ and f), then k, d_i, e_i, and B_i are the same as in this previous application; otherwise, $k = n + m$ and d_i, e_i, and $B_i \in \mathcal{B} \setminus \mathcal{A}$ are fresh.

Next, we show that the simplification rules of $\mathcal{BS}_{DL}$ are sound and correct, so they do not affect the soundness and completeness of $\mathcal{BS}$.

LEMMA 9 (Soundness). *In every $\mathcal{BS}_{DL}$ saturation, each application of a simplification rule is sound.*

Proof. Let $N_0, N_1, \ldots, N_n$ be a $\mathcal{BS}_{DL}$ saturation and I_0 a model of N_0. We prove the lemma by constructing a model I_s for each N_s with $1 \leq s \leq n$ inductively. Consider all possible cases for the inference producing N_s from N_{s-1}:

(Standard $\mathcal{BS}$ inferences) $I_s := I_{s-1}$ is clearly a model of N_s.

(Decomposition 1) If the predicate symbol A is reused in the inference, we set $I_s := I_{s-1}$; otherwise, we extend I_{s-1} to I_s by interpreting $A(x)$ exactly as $L\rho$. Obviously, I_s is a model of N_s.

(Decomposition 2) Analogous to Decomposition 1.

(Nominal Generation 1) If c_i are reused in the inference, we set $I_s := I_{s-1}$. If c_i are new and $\alpha(x)$ is true for all x, we extend I_{s-1} to I_s by

interpreting new symbols arbitrarily. Otherwise, $\alpha(x) \vee \bigvee_{i=1}^{n} [f(x)] \approx t_i$ ensures that, for those x for which $\alpha(x)$ is false in I_{s-1}, $f(x)$ has ℓ distinct values $o_1, \ldots, o_\ell$ in I_{s-1}, $1 \leq \ell \leq n$, so we extend I_{s-1} to I_s by interpreting c_i as o_i for $i \leq \ell$ and as o_1 for $i > \ell$.

Hence, if $\alpha(x)$ is true for all x, all conclusions are obviously true in I_s. Otherwise, c_i represent exactly those values $f(x)$ for which $\alpha(x)$ is false, so each c_i is interpreted in I_s as some t_j; therefore, all conclusions are true in I_s.

(Nominal Generation 2) If d_i, e_i, and B_i are reused in the inference, we set $I_s := I_{s-1}$. If d_i, e_i, and B_i are new and $\alpha(x)$ is true for all x, we extend I_{s-1} to I_s by interpreting d_i and e_i arbitrarily, and making B_i false everywhere. Otherwise, let x be such that $\alpha(x)$ is false in I_{s-1}. By Condition (ii), $\alpha'(x)$ is also false, so $x \approx g(f(x))$ holds in I_{s-1}. Moreover, $\alpha(x) \vee \bigvee_{i=1}^{n} [f(x)] \approx t_i \vee \bigvee_{i=1}^{m} x \approx c_i$ ensures that either $f(x)$ is equal to one of $t_1, \ldots, t_n$, or x is equal to one of $c_1, \ldots, c_m$. These two conditions imply that x is equal to one of $g(t_1), \ldots, g(t_n), c_1, \ldots, c_m$, so $\alpha(x)$ is false for exactly ℓ distinct domain elements $o_1, \ldots, o_\ell$, for $1 \leq \ell \leq n + m$. We extend I_{s-1} to I_s as follows: for $i \leq \ell$, we interpret d_i as o_i, e_i as $f(o_i)$ and make B_i true only for o_i; for $i > \ell$, we interpret d_i as o_1 and e_i as $f(o_1)$ and make B_i true only for o_1.

Hence, if $\alpha(x)$ is true for some x, then $B_i(x)$ is false, so all conclusions are obviously true in I_s. Otherwise, if $\alpha(x)$ is false, i exists such that d_i is equal to x, B_i holds only on x, and e_i is equal to $f(x)$. This makes the first three conclusions true in I_s; the fourth conclusion is true in I_s because of the premise. □

LEMMA 10 (Correctness). *All $\mathcal{BS}_{DL}$ simplification rules are correct.*

Proof. For each simplification rule with the premise $C \cdot \rho$ and conclusions $C_i \cdot \rho$, $1 \leq i \leq n$, ground substitution τ, and rewrite system $\mathcal{R}$, we need to show that, if $C \cdot \rho\tau$ is variable irreducible w.r.t. $\mathcal{R}$, then (i) $C_i \cdot \rho\tau$ are variable irreducible w.r.t. $\mathcal{R}$, (ii) $C_1\rho\tau, \ldots, C_n\rho\tau \models C\rho\tau$, and (iii) $C\rho\tau \succ C_i\rho\tau$. Property (i) is trivially satisfied for all simplification rules from Table VI, since each substitution position in $C_i \cdot \rho$ corresponds to a substitution position in $C \cdot \rho$. Next, we prove properties (ii) and (iii) for each rule. Let $u = x\tau$.

(Decomposition 1) The instance $C = D\rho\tau \vee L\rho\tau$ of the premise can be obtained by resolving the instances $E_1 = \neg A(u) \vee L\rho\tau$ and $E_2 = D\rho\tau \vee A(u)$ of the conclusions on $A(u)$. Furthermore, $D\rho\tau$ contains a term $h(u)$ by Condition (ii), and $h(u) \succ A(u)$ by Definition 8, so $D\rho\tau \succ A(u)$. Similarly, $L\rho\tau$ contains a term $f(u)$ by Condition (i), so $L\rho\tau \succ \neg A(u)$. Thus, $C \succ E_1$ and $C \succ E_2$.

(Decomposition 2) By Definition 8, $f(u) \approx c \succ B(u)$. Furthermore, by Condition (i), $D\rho$ contains either $h(x)$, but then $h(u) \succ \neg B(u)$,

or $D\rho$ contains $(\neg)A(x)$, but then $(\neg)A(u) \succ \neg B(u)$ by Definition 8. Hence, $D\rho\tau \succ \neg B(u)$, so the rest of the argument is analogous to Decomposition 1.

(Nominal Generation 1) The instance $C = \alpha(u) \vee \bigvee_{i=1}^{n} f(u) \approx t_i$ of the premise can be obtained by simultaneously paramodulating on each c_i from $E_i = \alpha(u) \vee \bigvee_{j=1}^{n} c_i \approx t_j$ into $D = \alpha(u) \vee \bigvee_{i=1}^{k} f(u) \approx c_i$. Furthermore, by Condition (i) of Nominal Generation 1, some $t = t_i$ is of the form $h(c)$, and, because $h(c) \succ c_i$, we have $f(u) \approx t \succ f(u) \approx c_i$, which implies $C \succ D$. Similarly, $f(u) \approx t_j \succ c_i \approx t_j$, so $C \succ E_i$.

(Nominal Generation 2) The instance

$$C = \alpha(u) \vee \bigvee_{i=1}^{n} f(u) \approx t_i \vee \bigvee_{i=1}^{m} u \approx c_i$$

of the premise can be obtained from the conclusions: first, paramodulate from $C_i = \neg B_i(u) \vee f(u) \approx e_i$ and from $D_i = \neg B_i(u) \vee u \approx d_i$ on e_i and d_i, respectively, into $E_i = \alpha(u) \vee \bigvee_{j=1}^{n} e_i \approx t_j \vee \bigvee_{j=1}^{m} d_i \approx c_j$; this produces $E_i' = \alpha(u) \vee \neg B_i(u) \vee \bigvee_{j=1}^{n} f(u) \approx t_j \vee \bigvee_{j=1}^{m} u \approx c_j$; then, resolve all E_i' with $F = \alpha(u) \vee \bigvee_{i=1}^{k} B_i(u)$ on $B_i(u)$ to obtain C. Furthermore, some $t = t_i$ is of the form $h(c)$ by Condition (i), so $h(c) \succ e_i$ implies $f(u) \approx t \succ f(u) \approx e_i$. Since $f(u) \succ \neg B_i(u)$, so $C \succ C_i$. Similarly, $f(u) \approx t \succ u \approx d_i$, so $C \succ D_i$. Finally, $f(u) \succ e_i$ and $f(u) \succ d_i$ imply $C \succ E_i$, and $f(u) \succ B_i(u)$ implies $C \succ F$. $\qquad\square$

5.2. Saturation of DL-Closures by $\mathcal{BS}_{DL}$

We now show that the inferences of $\mathcal{BS}_{DL}$, when applied to DL-closures, always produce a DL-closure.

LEMMA 11 (Closure under Inferences). *Let N be a set of DL-closures to which no $\mathcal{BS}_{DL}$ simplification rule is applicable. Then, an application of a $\mathcal{BS}_{DL}$ inference to N followed by exhaustive simplification produces a set of DL-closures.*

Proof. Before considering all possible inferences with closures from N, we consider the types of literals that can be eligible in each closure from N. Each closure of type 1–4 contains exactly one literal containing a function symbol; this literal is then eligible because it is maximal and no literal is selected. A closure of type 5 either contains a literal $x \not\approx t$, which is selected, or it contains a guard for x, which is selected. A closure of type 6 can contain a literal $x \not\approx c$, which is then selected. Otherwise, the closure must contain a literal $(\neg)B([f(x)])$: were this not the case, the closure would have the form $\alpha(x) \vee \bigvee [f(x)] \approx t_i \vee \bigvee x \approx c_i$;

Table VII. Eligible Literals in DL-Closures

1	$\alpha(x) \vee \boxed{(\neg)f(x) \approx g(x)}$	2	$\alpha(x) \vee \boxed{(\neg)f([g(x)]) \approx x}$
3	$\alpha(x) \vee \boxed{(\neg)A(f(x))}$	4	$\beta(x) \vee \boxed{(\neg)f(x) \approx c}$
5.1	$\beta(x) \vee \bigvee(\neg)x \approx t_i \vee \boxed{x \not\approx t}$	5.2	$\beta(x) \vee \boxed{\neg B(x)} \vee \bigvee x \approx t_i$

6.1	$\alpha(x) \vee \bigvee [f(x)] \approx t_i \vee \bigvee(\neg)x \approx c_i \vee \boxed{x \not\approx c}$
6.2	$\alpha(x) \vee \beta([f(x)]) \vee \boxed{(\neg)B([f(x)])} \vee \bigvee [f(x)] \approx t_i \vee \bigvee x \approx c_i$
7	$\alpha_1(x) \vee \boxed{\neg R(x,y)} \vee \alpha_2(y) \vee \bigvee_{i=1}^n f_i(x) \approx y$

8.1	$\boxed{\neg R(x,y)} \vee S(x,y)$	8.2	$\boxed{\neg R(x,y)} \vee S(y,x)$
9.1	$\alpha(x) \vee \boxed{R(x,f(x))}$	9.2	$\alpha(x) \vee \boxed{R(f(x),x)}$
10.1	$\beta(x) \vee \boxed{R(x,c)}$	10.2	$\beta(x) \vee \boxed{R(c,x)}$

11.1	$\boxed{(\neg)B(t)}$	11.2	$\boxed{(\neg)t_1 \approx t_2}$	11.3	$\boxed{(\neg)f(g(c)) \approx d}$
11.4	$\boxed{(\neg)R(c,d)}$	11.5	$\boxed{(\neg)R(c,f(d))}$	11.6	$\boxed{(\neg)R(f(c),d)}$

if some t_i is of the form $h(c)$, the closure would be simplified by Nominal Generation 1 or 2 (Condition (*ii*) of Nominal Generation 2 is satisfied because Condition (∗) holds for the premise); if all t_i are constants, the closure would be simplified by Decomposition 2 (Condition (*i*) is satisfied by a guard for x occurring in $\alpha(x)$). Since $B(f(x)) \succ f(x)$ and $B(f(x)) \succ g(c)$ by Definition 8, a literal of this form is eligible for inferences. The cases for the remaining closures are straightforward and are summarized in Table VII.

Next, we enumerate all $\mathcal{BS}_{DL}$-inferences between DL-closures and show that they result in DL-closures. With $[c1, c2] = [s] = [r1, r2, \ldots]$ we denote an inference between closures c1 and c2 resulting in closures $r1, r2, \ldots$, possibly by applying simplification s exhaustively. Cut, ground unit resolution, and closure subsumption ensure that ground literals occur only in unit closures; we call a combination of these inferences *splitting*.

Resolution inferences are possible only between closures of types 3, 5.2, 6.2, and 11.1 on unary literals; 9, 10, 11.4, 11.5, and 11.6 on positive binary literals; and 7, 8, 11.4, 11.5, and 11.6 on negative binary literals. Resolution with a premise of type 11 results in a ground closure, which is split into closures of type 11. The remaining resolution inferences are as follows: [3,3] = [5], [3,5.2] = [6], [3,6.2] = [6], [5.2,6.2] = [6], [6.2,6.2] = [6], [9.1,7] = [Decomposition 1] = [1,6], [9.2,7] = [Decomposition 1]

= [2,6], [9,8] = [9], [10.1,7] = [Decomposition 2, Splitting] = [4,5,11], [10.2,7] = [Splitting] = [5,11], [10,8] = [10].

Superposition inferences are possible from a nonground closure of type 1, 2, or 4, or from a ground closure of type 11.2 or 11.3, into a term $f(x)$ of 1, 3, 4, or 9 or into a term $f([g(x)])$ of 2 or into a ground (sub)term of 5.1, 6.1, 10, or 11. Note that superposition into or from a ground term does not increase the term depth, so the conclusion either becomes ground or is of the same type as the other premise. Therefore, we do not consider types 5.1, 6.1, 10, and 11 in the following case analysis.

Superposition from 1 into 1, 3, 4, or 9 produces closure of the latter type, since a function symbol f is just replaced by g: [1,1] = [1], [1,3] = [3], [1,4] = [4], [1,9]= [9]. Superposition from 1 into 2 produces $\alpha([g'(x)]) \vee \alpha'(x) \vee g([g'(x)]) \approx x$, which is simplified into types 2 and 6 using Decomposition 1.

Superposition from 2 into 1, 3, 4, or 9 instantiates the variable of the second premise to $[g(x)]$: [2,1] = [Decomposition 1] = [2,6], [2,3] = [6], [2,4] = [6], [2,9] = [Decomposition 1] = [9,6]. Superposition from 2 into 2 produces either a tautology that is deleted or a closure with a literal $x \not\approx x$ that is removed by reflexivity resolution and subsumption deletion, thus producing a closure of type 5.

Superposition from 4 into 1, 2, 3, 4, or 9 results in the following inferences: [4,1] = [4], [4,2] = [6], [4,3] = [Splitting] =[5,11], [4,4] = [Splitting] = [5,11], [4,9] = [10].

Reflexivity resolution inferences can be applied only to a closure of type 1, 5.1, 6.1, or 11.2. For 1 we obtain 5; in the remaining cases, the result is ground and is split into closures of type 11.

Factoring inferences are not applicable because duplicate literals are eagerly eliminated and closures with multiple equality literals are eagerly decomposed.

Condition (∗). Consider an inference producing a closure of type 6 with a literal $x \approx c_i$. Such an inference either is a superposition between 2 and 4, so the premise of type 2 validates Condition (∗) of the conclusion, or it has a premise of type 6, so $x \approx c_i$ in the conclusion stems from this premise. Hence, (∗) is satisfied for all conclusions of type 6.

Guards are preserved by all inferences because each premise contains a guard, and no inference involves a negative nonequational literal from all premises.

Simplification inferences always produce DL-closures: for our custom rules, this follows from Table VI, and for the remaining standard ones this is trivial. □

5.3. Termination and Complexity Analysis

We now show that each saturation of $\Gamma(KB)$ by $\mathcal{BS}_{DL}$ terminates. Assuming unary coding of numbers in number restrictions, the number of function symbols in $\Gamma(KB)$ is linear in $|KB|$.

LEMMA 12. *Let $\Gamma(KB) = N_0, N_1, \ldots, N_n$ be a path in a derivation by $\mathcal{BS}_{DL}$ from N_0. Then, the number of constants in each N_i is at most doubly exponential, and the number of closures in N_i is at most triply exponential in $|KB|$, for unary coding of numbers.*

Proof. Nominal Generation 1 and 2 introduce new constants at most once for a combination of $\alpha(x)$ and f. Other than the predicate symbols from $\Gamma(KB)$, $\alpha(x)$ can contain the predicate symbols A introduced by Decomposition 1, of which at most four are introduced for a pair of function symbols f and g. Hence, the number of disjunctions $\alpha(x)$ is at most exponential in $|KB|$, and so is the number of Nominal Generation inferences that introduce new constants. Furthermore, the premise of such an inference can involve all terms of the form c or $f(c)$ derived thus far, so the inference can increase the total number of constants only by a linear factor. Thus, the number of constants in N_i can be at most doubly exponential in $|KB|$.

Decomposition 2 introduces at most one predicate symbol B for a combination of f and c, and Nominal Generation 2 introduces at most one predicate symbol B_i for each e_i or d_i. Hence, the number of predicate symbols in N_i is at most doubly exponential in $|KB|$. Since each DL-closure contains at most one variable, the number of different literals is at most doubly exponential, so the number of DL-closures without repeated literals is at most triply exponential in $|KB|$.   □

THEOREM 13. *$\mathcal{BS}_{DL}$ decides satisfiability of a $\mathcal{SHOIQ}$ knowledge base KB in triply exponential time, for unary coding of numbers.*

Proof. Without loss of generality we can assume that an inference between two closures is performed at most once in a saturation. By Lemmas 7 and 11, each set of closures in a $\mathcal{BS}_{DL}$ saturation contains only DL-closures and is at most triply exponential in size by Lemma 12. Hence, all DL-closures are derived after at most triply exponential number of steps. Because simplification rules of $\mathcal{BS}_{DL}$ are sound and correct by Lemmas 9 and 10, the set of closures upon termination is saturated up to redundancy. Hence, $\Gamma(KB)$ and, by Lemma 6, KB are satisfiable if and only if the saturated set does not contain the empty closure.

Since $\mathcal{BS}_{DL}$ uses the cut rule, it is nondeterministic, so a straightforward complexity estimation gives us only a nondeterministic triply

exponential upper bound. This can be improved to a deterministic triply exponential bound as follows. The number of unit ground closures is at most doubly exponential, so the number of cut inferences performed on each branch of the saturation is at most doubly exponential. Hence, if we implement our procedure using backtracking, the number of all inferences is triply exponential. □

6. Discussion

In this section we discuss two important aspects of our algorithm.

6.1. Relationship with the Tableau Procedure

Nominal Generation 1 and 2 rules from Table VI are somewhat analogous to the Ro-rule from the tableau decision algorithm for $\mathcal{SHOIQ}$ [23]. In particular, the termination argument for tableau decision procedures is typically based on the so-called tree-model property: if a DL knowledge base is satisfiable, then it has a tree model [47]. Nominal nodes, however, can be connected in arbitrary ways. In fact, for the DL $\mathcal{SHOQ}$ (obtained from $\mathcal{SHOIQ}$ by disallowing inverse roles), we can consider only *forest models*, which consist of a "cloud" of arbitrarily interconnected nominals, and trees that emanate from nominals and whose nodes can have role arcs pointing to nominals. The size of these trees depends on the number of nominals (and which is bounded by the size of the input knowledge base), which can be exploited to obtain a decision procedure [22]. By adding inverse roles, however, we can restrict the number of links pointing to a nominal node, which in turn means that the predecessors of nominal nodes can also be interconnected arbitrarily; effectively, they behave just like nominals. The number of nominals is thus not bounded by the size of the input knowledge base, which invalidates the termination proof. To remedy the situation, given a nominal node that is an instance of a number restriction concept $\leqslant n\, S.C$, the Ro-rule guesses the actual number of neighboring nominals and introduces them explicitly. The number n is taken as an upper bound on the number of the introduced nominals. By obtaining the bound on the number of rule applications, one can establish termination of the algorithm.

The problems in the resolution setting are similar: because of inverse roles, nominals, and number restrictions, we can derive terms that play the role of new nominals. Nominal Generation rules then introduce new names for such terms, thus bounding the term growth. The main

difference from the tableau procedure, however, is that the application of these rules does not depend on some concrete model; rather, it is performed for all models that are described by the knowledge base. Furthermore, our rules are deterministic: they introduce new constants that are not necessarily different and that can be made equal later. Our procedure can thus be viewed as being "more deterministic."

6.2. Large Cardinality Restrictions and Resolution

The decision procedure for the DL $\mathcal{SHIQ}$ from [25, 26] is worst-case optimal (assuming unary coding of numbers), yet still practical. It is therefore somewhat disappointing that the procedure presented in this paper is not worst-case optimal. The high complexity of our procedure is due to a possibly doubly exponential number of constants introduced by Nominal Generation rules 1 and 2. In this section we analyze the situations in which this blowup occurs, and we provide an intuitive explanation why deriving an optimal resolution-based procedure is difficult, if not impossible.

Let KB be a knowledge base from Table VIII, which uses the well-known encoding of binary numbers by DL concepts. Every instance of the concept A is assigned a binary number $b_p\, b_{p-1} \ldots b_0$ with $b_i \in \{0,1\}$ using concepts A_i, $0 \leq i \leq p$, such that $b_i = 1$ iff A_i holds for the instance. Axioms TA1–9 ensure that the instances of A are arranged in a binary R-tree: each element of A has two R-successors, unless it is assigned a number $11 \ldots 1$ (TA1); and each element of A has at most one R-predecessor, unless it is assigned a number $00 \ldots 0$ (TA2). Axioms TA3-8 ensure that the number assigned to the children of a node in an R-tree is obtained by incrementing the number assigned to the parent: the last bit of the number always changes from the parent to the children (TA3,4), and every other bit changes if and only if the preceding bit changes from 1 to 0 (TA5-8). Axioms TA1–9 thus ensure that the number of elements at the kth level of this tree is at least 2^k; thus, the concept $A_p \sqcap \ldots \sqcap A_0$ that labels the leaves in the tree contains at least 2^{2^p} elements. Using a dual set of axioms TB1–TB9, we express in a similar way that the concept $B \sqcap B_q \sqcap \ldots \sqcap B_0$ contains at most 2^{2^q} elements.

Because of axiom T1, KB is satisfiable if and only if a set with 2^{2^p} elements can be embedded in a set with 2^{2^q} elements. Such combinatorial problems, commonly called the *pigeon hole principle*, are known to be hard for resolution [21]. On KB, our algorithm generates new nominals for all possible $\alpha(x) = \neg B(x) \vee (\neg)B_p(x) \vee \ldots \vee (\neg)B_0(x)$; intuitively, these represent constraints of the form $|B \sqcap (\neg)B_p \sqcap \ldots \sqcap (\neg)B_0| \leq 2^k$ by enumerating the set's elements using new constants. Although this

Table VIII. Expressing Large Cardinality Restrictions in $\mathcal{SHOIQ}$

TA1 $\neg(A_p \sqcap \cdots \sqcap A_0) \sqcap A \sqsubseteq \geqslant 2\,R.A$	TA3 $\quad A_0 \sqsubseteq \forall R.\neg A_0$
TA2 $\quad (A_p \sqcup \cdots \sqcup A_0) \sqcap A \sqsubseteq \leqslant 1\,R^-.A$	TA4 $\neg A_0 \sqsubseteq \forall R.A_0$
TA5 $\neg A_{i+1} \sqcap A_i \sqsubseteq \forall R.((\neg A_{i+1} \sqcap A_i) \sqcup (A_{i+1} \sqcap \neg A_i))$	TA7 $\neg A_{i+1} \sqcap \neg A_i \sqsubseteq \forall R.\neg A_{i+1}$
TA6 $\quad A_{i+1} \sqcap A_i \sqsubseteq \forall R.((A_{i+1} \sqcap A_i) \sqcup (\neg A_{i+1} \sqcap \neg A_i))$	TA8 $\quad A_{i+1} \sqcap \neg A_i \sqsubseteq \forall R.A_{i+1}$
TA9 $\quad \{c\} \sqsubseteq \neg A_p \sqcap \ldots \sqcap \neg A_0$	$i = 0 \ldots p-1$

TB1 $\neg(B_q \sqcap \cdots \sqcap B_0) \sqcap B \sqsubseteq \leqslant 2\,S.B$	TB3 $\quad B_0 \sqsubseteq \forall S.\neg B_0$
TB2 $\quad (B_q \sqcup \cdots \sqcup B_0) \sqcap B \sqsubseteq \geqslant 1\,S^-.B$	TB4 $\neg B_0 \sqsubseteq \forall S.B_0$
TB5 $\neg B_{i+1} \sqcap B_i \sqsubseteq \forall S.((\neg B_{i+1} \sqcap B_i) \sqcup (B_{i+1} \sqcap \neg B_i))$	TB7 $\neg B_{i+1} \sqcap \neg B_i \sqsubseteq \forall S.\neg B_{i+1}$
TB6 $\quad B_{i+1} \sqcap B_i \sqsubseteq \forall S.((B_{i+1} \sqcap B_i) \sqcup (\neg B_{i+1} \sqcap \neg B_i))$	TB8 $\quad B_{i+1} \sqcap \neg B_i \sqsubseteq \forall S.B_{i+1}$
TB9 $\neg B_q \sqcap \ldots \sqcap \neg B_0 \sqsubseteq \{c\}$	$i = 0 \ldots q-1$

T1 $\quad A \sqcap A_q \sqcap \ldots \sqcap A_0 \sqsubseteq B \sqcap B_p \sqcap \ldots \sqcap B_0$

TA1–9 express $|A \sqcap A_p \sqcap \ldots \sqcap A_0| \geq 2^{2^p}$; TB1–9 express $|B \sqcap B_q \sqcap \ldots \sqcap B_0| \leq 2^{2^q}$

observation does not prove that an optimal resolution-based procedure for $\mathcal{SHOIQ}$ cannot exist, it demonstrates the weakness of resolution in dealing with large numbers and suggests that explicit algebraic operations on numbers might be necessary for optimal complexity.

Worst-case complexity does not, however, say much about the typical case. The example from Table VIII is problematic because it succinctly encodes binary numbers. Many applications do not require such combinatorial reasoning. In such cases, our algorithm should not introduce too many new constants. In fact, new nominals are generated only due to terms $g(c)$, which can result only from an interaction between inverse roles, number restrictions, and nominals. If these constructs are not used simultaneously, our algorithm is similar to the algorithm for the DL $\mathcal{SHIQ}$ from [25], and it runs in exponential time. Thus, our algorithm exhibits "pay-as-you-go" behavior.

7. Conclusion

In this paper, we presented a resolution-based procedure for deciding satisfiability of a $\mathcal{SHOIQ}$ knowledge base KB running in triply exponential time. Extending the existing decision procedures for DLs such as $\mathcal{SHIQ}$ is not trivial because of an intricate interaction between nominals, inverse roles, and number restrictions. We base our procedure on basic superposition [6, 36], which we additionally extend with new simplification rules that break apart dangerous clauses and thus ensure termination. We believe that our explanations provide useful guidance

for the construction of resolution-based procedures for other fragments of first-order logic.

The main challenge for future research is to derive a practical, but worst-case optimal decision procedure for $\mathcal{SHOIQ}$. The problems outlined in Section 6 might be addressed by integrating algebraic reasoning directly into resolution, as was done, for example, for tableau calculi [20].

References

1. Armando, A., S. Ranise, and M. Rusinowitch: 2001, 'Uniform Derivation of Decision Procedures by Superposition'. In: *Proc. of the 15th Int. Workshop on Computer Science Logic (CSL' 01)*, Vol. 2142 of *LNCS*. Paris, France, pp. 549–563.

2. 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.

3. Baader, F. and T. Nipkow: 1998, *Term Rewriting and All That*. Cambridge University Press.

4. Bachmair, L. and H. Ganzinger: 1994, 'Rewrite-based Equational Theorem Proving with Selection and Simplification'. *Journal of Logic and Computation* **4**(3), 217–247.

5. Bachmair, L. and H. Ganzinger: 2001, 'Resolution Theorem Proving'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. I. Elsevier Science, Chapt. 2, pp. 19–99.

6. Bachmair, L., H. Ganzinger, C. Lynch, and W. Snyder: 1995, 'Basic Paramodulation'. *Information and Computation* **121**(2), 172–192.

7. Bachmair, L., H. Ganzinger, and U. Waldmann: 1993, 'Superposition with Simplification as a Decision Procedure for the Monadic Class with Equality'. In: *Proc. of the Third Kurt Gödel Colloquium on Computational Logic and Proof Theory (KGC '93)*, Vol. 713 of *LNCS*. Brno, Czech Republic, pp. 83–96.

8. Baumgartner, P. and R. A. Schmidt: 2006, 'Blocking and Other Enhancements for Bottom-Up Model Generation Methods'. In: *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *LNCS*. Seattle, WA, USA, pp. 125–139.

9. Bergamaschi, S., S. Castano, M. Vincini, and D. Beneventano: 2001, 'Semantic Integration of Heterogeneous Information Sources'. *Data & Knowledge Engineering* **36**(3), 215–249.

10. Borgida, A.: 1996, 'On the Relative Expressiveness of Description Logics and Predicate Logics'. *Artificial Intelligence* **82**(1–2), 353–367.

11. Calvanese, D., M. Lenzerini, and D. Nardi: 1999, 'Unifying Class-Based Representation Formalisms'. *Journal of Artificial Intelligence Research (JAIR)* **11**, 199–240.

12. de Nivelle, H.: 1995, 'Ordering Refinements of Resolution'. Ph.D. thesis, Technische Universiteit Delft.

13. de Nivelle, H.: 2000, 'Deciding the E-plus class by an a posteriori, liftable order'. *Annals of Pure and Applied Logic* **88**(1), 219–232.

14. de Nivelle, H. and M. de Rijke: 2003, 'Deciding the Guarded Fragments by Resolution'. *Journal of Symbolic Computation* **35**, 21–58.

15. de Nivelle, H. and I. Pratt-Hartmann: 2001, 'A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality'. In: *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, Vol. 2083 of *LNAI*. Siena, Italy, pp. 211–225.

16. Fermüller, C., T. Tammet, N. Zamov, and A. Leitsch: 1993, *Resolution Methods for the Decision Problem*, Vol. 679 of *LNAI*. Springer.

17. Fermüller, C. G., A. Leitsch, U. Hustadt, and T. Tammet: 2001, 'Resolution Decision Procedures'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. II. Elsevier Science, Chapt. 25, pp. 1791–1849.

18. Ganzinger, H. and H. de Nivelle: 1999, 'A Superposition Decision Procedure for the Guarded Fragment with Equality'. In: *Proc. of the 14th IEEE Symposium on Logic in Computer Science (LICS '99)*. Trento, Italy, pp. 295–305.

19. Goasdoué, F. and M.-C. Rousset: 2004, 'Answering Queries using Views: A KRDB Perspective for the Semantic Web'. *ACM Transactions on Internet Technology* **4**(3), 255–288.

20. Haarslev, V., M. Timmann, and R. Möller: 2001, 'Combining Tableaux and Algebraic Methods for Reasoning with Qualified Number Restrictions'. In: *Proc. of the 2001 Int. Workshop on Description Logics (DL 2001)*, Vol. 49 of *CEUR Workshop Proceedings*. Stanford, CA, USA.

21. Haken, A.: 1985, 'The Intractability of Resolution'. *Theorerical Computer Science* **39**, 297–308.

22. Horrocks, I. and U. Sattler: 2001, 'Ontology Reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ Description Logic'. In: B. Nebel (ed.): *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. Seattle, WA, USA, pp. 199–204.

23. Horrocks, I. and U. Sattler: 2005, 'A Tableaux Decision Procedure for $\mathcal{SHOIQ}$'. In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Edinburgh, UK, pp. 448–453.

24. Horrocks, I., U. Sattler, and S. Tobies: 2000, 'Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$'. In: *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, Vol. 1831 of *LNAI*. Pittsburgh, PA, USA, pp. 482–496.

25. Hustadt, U., B. Motik, and U. Sattler: 2004, 'Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs'. In: *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*. Whistler, Canada, pp. 152–162.

26. Hustadt, U., B. Motik, and U. Sattler: 2005, 'A Decomposition Rule for Decision Procedures by Resolution-based Calculi'. In: *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.

27. Hustadt, U. and R. A. Schmidt: 1999, 'Maslov's Class K Revisited'. In: *Proc. of the 16th Int. Conf. on Automated Deduction (CADE-16)*, Vol. 1632 of *LNAI*. Trento, Italy, pp. 172–186.

28. Joyner, W. H.: 1976, 'Resolution Strategies as Decision Procedures'. *Journal of the ACM* **23**(3), 398–417.

29. Kazakov, Y.: 2006, 'Saturation-Based Decision Procedures for Extensions of the Guarded Fragment'. Ph.D. thesis, Univesität des Saarlandes, Germany.

30. Kazakov, Y. and H. de Nivelle: 2004, 'A Resolution Decision Procedure for the Guarded Fragment with Transitive Guards'. In: *Proc. of 2nd Int. Joint Conf. on Automated Reasoning (IJCAR 2004)*, Vol. 3097 of *LNAI*. Cork, Ireland, pp. 122–136.

31. Kazakov, Y. and B. Motik: 2006, 'A Resolution-Based Decision Procedure for $\mathcal{SHOIQ}$'. In: *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *LNCS*. Seattle, WA, USA, pp. 662–667.

32. Leitsch, A.: 1993, 'Deciding clause classes by semantic clash resolution'. *Fundamenta Informaticae* **18**, 163–182.

33. Levy, A. Y., D. Srivastava, and T. Kirk: 1995, 'Data Model and Query Evaluation in Global Information Systems'. *Journal of Intelligent Information Systems* **5**(2), 121–143.

34. Motik, B.: 2006, 'Reasoning in Description Logics using Resolution and Deductive Databases'. Ph.D. thesis, Univesität Karlsruhe, Germany.

35. Motik, B. and U. Sattler: 2006, 'A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes'. In: *Proc. of the 13th Int. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006)*, Vol. 4246 of *LNCS*. Phnom Penh, Cambodia, pp. 227–241.

36. Nieuwenhuis, R. and A. Rubio: 1995, 'Theorem Proving with Ordering and Equality Constrained Clauses'. *Journal of Symbolic Computation* **19**(4), 312–351.

37. Nieuwenhuis, R. and A. Rubio: 2001, 'Paramodulation-Based Theorem Proving'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. I. Elsevier Science, Chapt. 7, pp. 371–443.

38. Nivelle, H. D., R. A. Schmidt, and U. Hustadt: 2000, 'Resolution-Based Methods for Modal Logics'. *Logic Journal of the IGPL* **8**(3), 265–292.

39. Pacholski, L., W. Szwast, and L. Tendera: 2000, 'Complexity Results for First-Order Two-Variable Logic with Counting'. *SIAM Journal on Computing* **29**(4), 1083–1117.

40. Patel-Schneider, P. F., P. Hayes, and I. Horrocks: 2004, 'OWL Web Ontology Language: Semantics and Abstract Syntax, W3C Recommendation'. http://www.w3.org/TR/owl-semantics/.

41. Peltier, N.: 2001, 'On the decidability of the PVD class with equality'. *Logic Journal of the IGPL* **9**(4), 601–624.

42. Plaisted, D. A. and S. Greenbaum: 1986, 'A Structure-Preserving Clause Form Translation'. *Journal of Symbolic Logic and Computation* **2**(3), 293–304.

43. Pratt-Hartmann, I.: 2005, 'Complexity of the Two-Variable Fragment with Counting Quantifiers'. *Journal of Logic, Language and Information* **14**(3), 369–395.

44. Schmidt, R. A. and U. Hustadt: 2003, 'A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae'. In: *Proc. of the 19th Int. Conf. on Automated Deduction (CADE-19)*, Vol. 2741 of *LNAI*. Miami Beach, FL, USA, pp. 412–426.

45. Tammet, T.: 1992, 'Resolution Methods for Decision Problems and Finite-Model Building'. Ph.D. thesis, Göteborg University, Sweden.

46. Tobies, S.: 2001, 'Complexity Results and Practical Algorithms for Logics in Knowledge Representation'. Ph.D. thesis, RWTH Aachen, Germany.

47. Vardi, M. Y.: 1996, 'Why Is Modal Logic So Robustly Decidable?'. In: N. Immerman and P. Kolaitis (eds.): *Proc. of a DIMACS Workshop on Descriptive Complexity and Finite Models*, Vol. 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Princeton University, USA, pp. 149–184.

48. Weidenbach, C.: 2001, 'Combining Superposition, Sorts and Splitting'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*, Vol. II. Elsevier Science, Chapt. 27, pp. 1965–2013.