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

Yevgeny Kazakov and Boris Motik

University of Manchester, 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. Applying existing techniques for deriving saturation-based decision procedures to $\mathcal{SHOIQ}$ is not straightforward due to 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.

1 Introduction

Description logics (DLs) are a family of knowledge representation formalisms [2] that have been applied in numerous areas of computer science, such as information integration and ontology modeling. In particular, the DL $\mathcal{SHOIQ}$ provides a logical foundation for the Web Ontology Language (OWL)—the language standardized by the W3C for building ontologies in the Semantic Web. Thus, to implement advanced Semantic Web applications based on OWL-DL, practical reasoning procedures for $\mathcal{SHOIQ}$ are required.

It is known that $\mathcal{SHOIQ}$ can be embedded into $\mathcal{C}^2$ [21]—the two-variable fragment of first-order logic with counting quantifiers. Furthermore, $\mathcal{C}^2$ is decidable in NExpTime [18] (this result was recently sharpened to allow for binary coding of numbers [19]). However, all known decision procedures for $\mathcal{C}^2$ use a rather blind "guess-and-check" approach, which is unlikely to be suitable for practical purposes. Rather, a practical algorithm should be goal-oriented, using the input problem to guide the search.

Designing such a procedure for $\mathcal{SHOIQ}$ has proved to be a nontrivial task. Namely, this logic provides for inverse roles, number restrictions, and *nominals*—concepts with a bounded number of elements. The intricate interaction between these constructs makes extending existing (tableau-based) procedures difficult. Only recently, a goal-directed tableau-based procedure was presented in [10]; it uses a nondeterministic guess on the size of nominals to ensure termination.

In this paper, we present an alternative reasoning procedure based on resolution. $\mathcal{SHOIQ}$ is a hard logic, so it is not obvious which reasoning method is most suitable for practice. Rather, comparing different methods and identifying which ones are suitable for which types of problems can give crucial insights

into building practical reasoning systems. Furthermore, this procedure is based on the same principles as the procedure for a weaker logic $\mathcal{SHIQ}$ [11], which was implemented in a new reasoning system KAON2,[1] and has shown promising results for answering queries over large data sets.

To obtain an algorithm for $\mathcal{SHOIQ}$, we face problems analogous to those encountered in constructing the tableau algorithm from [10]. 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 using novel *simplification* rules, which rename complex terms with simpler ones based on their semantic meaning.

2 Preliminaries

Description Logic $\mathcal{SHOIQ}$. Given a set of role names N_R, a *role* is either some $R \in N_R$ or an *inverse role* R^- for $R \in N_R$. An *RBox* $KB_\mathcal{R}$ is a finite set of role inclusion axioms $R \sqsubseteq S$ and transitivity axioms $\mathsf{Trans}(R)$, for R and S roles. For $R \in N_R$, we set $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$, and assume that $R \sqsubseteq S \in KB_\mathcal{R}$ ($\mathsf{Trans}(R) \in KB_\mathcal{R}$) implies $\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \in KB_\mathcal{R}$ ($\mathsf{Trans}(\mathsf{Inv}(R)) \in KB_\mathcal{R}$). A role R is said to be *simple* if $\mathsf{Trans}(S) \notin KB_\mathcal{R}$ for each $S \sqsubseteq^* R$, where $\sqsubseteq^*$ is the reflexive-transitive closure of $\sqsubseteq$.

For N_C a set of *concept names* and N_I a set of *individuals*, 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 $A \in N_C$, 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 $KB_\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$. An ABox $KB_\mathcal{A}$ is a finite set 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 KB is a triple $(KB_\mathcal{R}, KB_\mathcal{T}, KB_\mathcal{A})$. With $|KB|$ we denote the number of symbols needed to encode KB using unary coding of numbers. KB is given semantics by translating it into first-order logic using the operator π defined in Table 1. The main inference problem in $\mathcal{SHOIQ}$ is checking satisfiability of KB, or, equivalently, of $\pi(KB)$.

Basic Superposition Calculus. We assume familiarity with standard notions of resolution theorem proving [4] and term rewriting [3]. For a term t, $t|_p$ denotes the subterm of t at the position p, and $t[s]_p$ denotes the replacement of $t|_p$ in t with a term s. We encode literals $(\neg)A$ as $(\neg)A \approx \mathsf{tt}$ in a multi-sorted setting, so $\approx$ is the only predicate symbol. We do not distinguish $(\neg)s \approx t$ from $(\neg)t \approx s$. For a literal L, with $\overline{L}$ we denote a literal obtained from L by flipping its sign.

Basic superposition ($\mathcal{BS}$) [5, 15] is a calculus for equational theorem proving. Its inference rules work with *closures*, which consist of (i) a *skeleton* clause C and (ii) a *substitution* σ. A closure is written as $C \cdot \sigma$ and it is semantically interpreted as the clause $C\sigma$; it is *ground* if $C\sigma$ is ground. The *empty closure* is denoted by $\square$. A closure can be conveniently represented by *marking* with

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

Table 1. Semantics of $\mathcal{SHOIQ}$ by Mapping to FOL

Translating Concepts to FOL
$\pi_y(\top, X) = \top$
$\pi_y(\bot, X) = \bot$
$\pi_y(A, X) = A(X)$
$\pi_y(\{a\}, X) = X \approx a$
$\pi_y(\neg C, X) = \neg\pi_y(C, X)$
$\pi_y(C \sqcap D, X) = \pi_y(C, X) \wedge \pi_y(D, X)$
$\pi_y(C \sqcup D, X) = \pi_y(C, X) \vee \pi_y(D, X)$
$\pi_y(\exists R.C, X) = \exists y : R(X, y) \wedge \pi_x(C, y)$
$\pi_y(\forall R.C, X) = \forall y : R(X, y) \rightarrow \pi_x(C, y)$
$\pi_y(\geqslant n\, S.C, X) = \exists y_1, \ldots, y_n : \bigwedge_{i=1}^{n} [S(X, y_i) \wedge \pi_x(C, y_i)] \wedge \bigwedge_{1 \leq i < j \leq n} y_i \not\approx y_j$
$\pi_y(\leqslant n\, S.C, X) = \forall y \exists y_1, \ldots, y_n : [S(X, y) \wedge \pi_x(C, y)] \rightarrow \bigvee_{i=1}^{n} y \approx y_i$
Translating Axioms to FOL
$\pi(C \sqsubseteq D) = \forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$
$\pi(R \sqsubseteq S) = \forall x, y : R(x, y) \rightarrow S(x, y)$
$\pi(\mathsf{Trans}(R)) = \forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
$\pi(C(a)) = \pi_y(C, a)$
$\pi(R(a, b)) = R(a, b)$
$\pi(a \circ b) = a \circ b$ for $\circ \in \{\approx, \not\approx\}$
$\pi(KB) = \bigwedge_{R \in N_R} \forall x, y : [R(x, y) \leftrightarrow R^-(y, x)] \wedge \bigwedge_{\alpha \in KB_{\mathcal{T}} \cup KB_{\mathcal{R}} \cup KB_{\mathcal{A}}} \pi(\alpha)$
π_x is obtained by simultaneously substituting in the definition of π_y all $x_{(i)}$ for all $y_{(i)}$, respectively, and π_y for π_x. X is a meta-variable and is substituted by the actual variable.

[ ] the terms from $C\sigma$ occurring at variable positions of C. Any position at or beneath a marked position is called a *substitution position*. For example, $(P(x) \vee z \approx b) \cdot \{x \mapsto f(y), z \mapsto g(b)\}$ can be written as $P([f(y)]) \vee [g(b)] \approx b$.

The $\mathcal{BS}$ calculus is parameterized with an admissible ordering on terms and a selection function. An *admissible ordering on terms* $\succ$ is a *reduction ordering* total on ground terms. The term ordering is extended to literals by identifying a literal $s \approx t$ with a multiset $\{\{s\}, \{t\}\}$ and a literal $s \not\approx t$ with a multiset $\{\{s, t\}\}$, and by comparing these multisets by a two-fold multiset extension of the term ordering $\succ$; we denote the literal ordering also with $\succ$. A *selection function* selects an arbitrary set of negative literals in each clause.

A literal $L\cdot\sigma$ is (strictly) maximal w.r.t. a closure $C\cdot\sigma$ if $L'\sigma \succ L\sigma$ ($L'\sigma \succeq L\sigma$) for no $L' \in C$. A literal $L \cdot \sigma$ is *(strictly) eligible* in $(C \vee L) \cdot \sigma$ if either (i) no literal is selected in $(C \vee L) \cdot \sigma$ and $L \cdot \sigma$ is (strictly) maximal w.r.t. $C \cdot \sigma$, or (ii) $L \cdot \sigma$ is selected in $(C \vee L) \cdot \sigma$. The inference rules of $\mathcal{BS}$ are presented in Table 2. Note that the standard resolution and factoring inference rules can be viewed as "macros," combining negative superposition and equality factoring, respectively, with reflexivity resolution.

We next present the redundancy criteria for closures. Let $\mathcal{R}$ be a ground and convergent rewrite system, and $C \cdot \sigma$ a ground closure. A variable x in the skeleton C of $C \cdot \sigma$ 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

Table 2. Inference Rules of the $\mathcal{BS}$ Calculus

Positive superposition: $\dfrac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \approx v) \cdot \rho}{(C \vee D \vee w[t]_p \approx v) \cdot \theta}$	(i) $\sigma = \mathsf{MGU}(s\rho, w\rho\vert_p)$ and $\theta = \rho\sigma$; (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$; (iii) $(s \approx t) \cdot \theta$ is strictly eligible; (iv) $(w \approx v) \cdot \theta$ is strictly eligible; (v) $s\theta \approx t\theta \not\succeq w\theta \approx v\theta$; (vi) $w\vert_p$ is not a variable.
Negative superposition: $\dfrac{(C \vee s \approx t) \cdot \rho \quad (D \vee w \not\approx v) \cdot \rho}{(C \vee D \vee w[t]_p \not\approx v) \cdot \theta}$	(i) $\sigma = \mathsf{MGU}(s\rho, w\rho\vert_p)$ and $\theta = \rho\sigma$; (ii) $t\theta \not\succeq s\theta$ and $v\theta \not\succeq w\theta$; (iii) $(s \approx t) \cdot \theta$ is strictly eligible; (iv) $(w \not\approx v) \cdot \theta$ is eligible; (v) $w\vert_p$ is not a variable.
Reflexivity resolution: $\dfrac{(C \vee s \not\approx t) \cdot \rho}{C \cdot \theta}$	(i) $\sigma = \mathsf{MGU}(s\rho, t\rho)$ and $\theta = \rho\sigma$; (ii) $(s \not\approx t) \cdot \theta$ is eligible.
Equality factoring: $\dfrac{(C \vee s \approx t \vee s' \approx t') \cdot \rho}{(C \vee t \not\approx t' \vee s' \approx t') \cdot \theta}$	(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.

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. a rewrite system $\mathcal{R}$.

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 $C\sigma\tau \succ C_i$, where $\succ$ is the multiset extension of the literal ordering to closures.

We extend basic superposition with several *simplification rules*, which can simplify a closure set $N \cup \{C \cdot \rho\}$ to $N \cup \{C_1 \cdot \rho, \ldots, C_n \cdot \rho\}$. Such application of a simplification rule is *sound* if it preserves satisfiability; it is *correct* if the closure $C \cdot \rho$ is redundant w.r.t. $N \cup \{C_1 \cdot \rho, \ldots, C_n \cdot \rho\}$. In [5] the authors present several sound and correct simplification rules, such as elimination of duplicate literals, tautology deletion, closure subsumption, and elimination of marked positions—that is, replacing $C \cdot \sigma$ with $C\rho \cdot \theta$ such that $\sigma = \rho\theta$.

The following two simplification rules can be used to split off ground literals in closures: *cut* nondeterministically derives $L \cdot \{\}$ or $\overline{L} \cdot \{\}$ for a ground literal L, and *ground unit resolution* simplifies a closure $C \cdot \rho \vee L \cdot \rho$ into $C \cdot \rho$ if the closure set contains a ground closure $L' \cdot \theta$ such that $\overline{L\rho} = L'\theta$.

Let N_0 be a set of closures of the form $C \cdot \{\}$, and let N_∞ be obtained by a fair saturation of N_0 by $\mathcal{BS}$ up to redundancy. Then, N_0 is unsatisfiable if and only if N_∞ contains the empty closure.

3 Preprocessing

We split our decision procedure into a *preprocessing phase*, which converts a $\mathcal{SHOIQ}$ knowledge base KB into a set of closures of a certain type, and a *saturation phase*, which checks satisfiability of the closure set using $\mathcal{BS}$. In the rest of this section, we present the preprocessing phase in detail.

Table 3. Closure Types after Preprocessing

Axiom	Closure
$R = \mathsf{Inv}(S)$	1. $\neg R(x,y) \vee S(y,x)$ and $\neg S(x,y) \vee R(y,x)$
$R \sqsubseteq S$	2. $\overline{\neg R(x,y)} \vee S(x,y)$
$L_1 \sqsubseteq \geqslant n\, R.L_2$	3. $\overline{L_1(x)} \vee R(x, f_i(x))$
	4. $\overline{L_1(x)} \vee L_2(f_i(x))$
	5. $\overline{L_1(x)} \vee f_i(x) \not\approx f_j(x)$ $\quad 1 \leq i < j \leq n$
$\top \sqsubseteq \bigsqcup L_i$	6. $\bigvee L_i(x)$
$L \sqsubseteq \{c\}$	7. $\overline{L(x)} \vee x \approx c$
$L_1 \sqsubseteq \leqslant n\, R.L_2$	8. $\overline{L_1(x)} \vee \neg R(x,y) \vee \overline{L_2(y)} \vee \bigvee_{i=1}^{n} f_i(x) \approx y$
$L(c)$	9. $L(c)$
$R(c,d)$	10. $R(c,d)$
$c \approx d$	11. $c \approx d$
$c \not\approx d$	12. $c \not\approx d$

Note: $L_{(i)}$ are of the form A or $\neg A$ for A an atomic concept. $L_1 \sqsubseteq \exists R.L_2$ and $L_1 \sqsubseteq \forall R.L_2$ are translated as $L_1 \sqsubseteq \geqslant 1\, R.L_2$ and $L_1 \sqsubseteq \leqslant 0\, R.\overline{L_2}$, respectively.

Eliminating Transitivity Axioms. It is well-known that deciding a logic with transitivity axioms by means of saturation calculi is difficult, and that it requires advanced techniques [13]. Therefore, we eliminate transitivity axioms by polynomially encoding KB into an equisatisfiable knowledge base $\Omega(KB)$ without such axioms. Roughly speaking, a transitivity axiom $\mathsf{Trans}(S)$ is replaced with axioms $\forall R.C \sqsubseteq \forall S.(\forall S.C)$, for each R with $S \sqsubseteq^* R$ and C is a "relevant" concept from KB. For more details, please see [14, Section 5.2]. (The latter result considers only $\mathcal{SHIQ}$; for $\mathcal{SHOIQ}$, the encoding is the same, and extending the correctness proof is trivial.) Similar encodings were presented in [21, 20].

Translation into Closures. Next, we simplify the TBox axioms of $\Omega(KB)$ by introducing new concept names for nonatomic subconcepts. For example, we simplify the axiom $C \sqsubseteq \exists R.\exists S.A$ by introducing a new concept Q and by replacing this axiom with $C \sqsubseteq \exists R.Q$ and $Q \sqsubseteq \exists S.A$. This transformation is analogous to the *structural transformation* [17]; for details, please see [14, Section 5.3.1].

For unary coding of numbers in number restrictions, this transformation is polynomial, and it preserves satisfiability. Furthermore, it produces axioms containing at most one nonatomic concept. Such axioms are converted into closures by translating them into first-order logic using the operator π from Table 1, skolemizing the existential quantifiers, and translating the result into conjunctive normal form. We denote the resulting set of closures with $\Xi(KB)$. Table 3 shows the closures that are produced by different types of axioms.

Introduction of Guards. Using nominals, one can restrict the cardinality of the interpretation domain, which makes it possible to derive a closure of the form $x \approx a_1 \vee \cdots \vee x \approx a_n$. Such closures can cause problems, since the variable x is *unshielded* (that is, x does not occur in the closure as a proper subterm). This allows for superposition inferences from x, which are prolific since x unifies with any term. We avoid this problem using the following transformation.

Definition 1. *For a closure $C\cdot\rho$, a variable x of $C\rho$ is* guarded *if it occurs in $C\rho$ in a negative nonequational literal, called a* guard *for x. Let KB be a $\mathcal{SHOIQ}$ knowledge base and T a predicate not occurring in $\Xi(KB)$. Then, $\Gamma(KB)$ is the smallest set such that (*i*) for each closure $C\cdot\rho\in\Xi(KB)$, $\Gamma(KB)$ contains the closure $\neg T(x_1)\vee\cdots\vee\neg T(x_n)\vee C\cdot\rho$, 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 function symbol f occurring in $\Xi(KB)$, $\Gamma(KB)$ contains the closure $\neg T(x)\vee T(f(x))$.*

Lemma 1. *$\Xi(KB)$ is satisfiable if and only if $\Gamma(KB)$ is satisfiable.*

Proof. ($\Rightarrow$) Let I be a model of $\Xi(KB)$, and I' an interpretation obtained by making $T(x)$ 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 $\Xi(KB)$. □

4 Saturating Closures by Basic Superposition

After preprocessing, our algorithm continues by saturating the set of closures $\Gamma(KB)$ by basic superposition. To prove that the saturation terminates, we apply the approach used in most existing resolution-based procedures [12, 6, 7, 16, 1]. We define a class of closures $\mathcal{N}_{DL}$ and demonstrate the following properties: (*i*) $\mathcal{N}_{DL}$ contains the closures obtained by translating a $\mathcal{SHOIQ}$ knowledge base KB, (*ii*) applying an inference rule of $\mathcal{BS}$ to closures from $\mathcal{N}_{DL}$ produces a closure in $\mathcal{N}_{DL}$, and (*iii*) $\mathcal{N}_{DL}$ contains finitely many closures for a finite signature. For a fixed signature, Conditions (*i*)–(*iii*) ensure that $\mathcal{BS}$ produces only finitely many closures from $\mathcal{N}_{DL}$ in a saturation. However, as we discuss in the following subsection, to coerce $\mathcal{BS}$ into producing closures of a restricted syntactic structure, we introduce several novel *simplification* rules. These rules can extend the signature, so, to ensure termination, we additionally show that (*iv*) the signature is extended only a finite number of times in a saturation.

4.1 The Problems in Ensuring Termination for $\mathcal{SHOIQ}$

It is well-known that reasoning in $\mathcal{SHOIQ}$ is difficult due to a subtle interaction involving nominals, number restrictions, and inverse roles. This interaction makes it difficult to ensure termination of $\mathcal{BS}$ on $\Xi(KB)$. We demonstrate these problems next on an example and sketch our solution.

Let KB be a $\mathcal{SHOIQ}$ knowledge base containing axioms T1–T4 from Table 4. The preprocessing step, when applied to these axioms, produces the closures (1)–(9) shown on the far right; furthermore, saturation of (1)–(9) by basic superposition produces closures (10)–(18). Note that (18) is similar in structure to (11): (18) just contains two literals $f_i(x)\approx g_1(c)$ and $f_i(x)\approx g_2(c)$ instead of just one literal $f_i(x)\approx c$; additionally, (18) contains $\neg T(x)$. It is easy to see that all inferences with (11) can be repeated for (18), and that this would produce

Table 4. Example of Termination Problems

Knowledge base KB and the closures obtained after preprocessing:			
T1. $O \sqsubseteq \{c\}$	$\Rightarrow$	$\Rightarrow$	(1) $\neg O(x) \vee x \approx c$
T2. $\top \sqsubseteq \exists R_1.\top \sqcup \exists R_2.\top$	$\Rightarrow$ $\top \sqsubseteq U_1 \sqcup U_2$	$\Rightarrow$	(2) $\neg \overline{T(x)} \vee U_1(x) \vee U_2(x)$
	$\Rightarrow$ $U_i \sqsubseteq \exists R_i.\top, \quad i = 1,2$	$\Rightarrow$	(3) $\neg \overline{U_i(x)} \vee \underline{R_i(x, f_i(x))}$
T3. $\top \sqsubseteq \leqslant 1\, R_i^-.\top,\ i = 1,2$	$\Rightarrow$	$\Rightarrow$	(4) $\neg \underline{R_i^-(x,y)} \vee g_i(x) \approx y$
	$\Rightarrow$ inverses of $R_i,\ i = 1,2$	$\Rightarrow$	(5) $\neg \overline{R_i(x,y)} \vee R_i^-(y,x)$
T4. $O \sqsubseteq \forall R_i.O, \quad i = 1,2$	$\Rightarrow$	$\Rightarrow$	(6) $\neg \overline{O(x)} \vee \neg \underline{R_i(x,y)} \vee O(y)$
	introduction of guards	$\Rightarrow$	(7) $\underline{T(c)}$
		$\Rightarrow$	(8) $\neg \overline{T(x)} \vee \underline{T(f_i(x))}$
		$\Rightarrow$	(9) $\neg T(x) \vee \underline{T(g_i(x))}$
Saturation of (1)–(9):			
[Resolving 3 with 6]:	(10) $\neg U_i(x) \vee \neg O(x) \vee \underline{O([f_i(x)])}$		
[Resolving 10 with 1]:	**(11)** $\boldsymbol{\neg U_i(x) \vee \neg O(x)} \vee \underline{[f_i(x)]} \approx c$		
[Superposing 11 into 3]:	(12) $\neg U_i(x) \vee \neg O(x) \vee \underline{R_i(x,c)}$		
[Resolving 12 with 5]:	(13) $\neg U_i(x) \vee \neg O(x) \vee \underline{R_i^-(c,x)}$		
[Resolving 13 with 4]:	(14) $\neg \underline{U_i(x)} \vee \neg O(x) \vee \underline{x \approx g_i(c)}$		
[Resolving 2 with 8]:	(15) $\neg \overline{T(x)} \vee U_1([f_i(x)]) \vee U_2([f_i(x)])$		
[Resolving 15 with 14]:	(16) $\neg T(x) \vee \underline{U_2([f_i(x)])} \vee \neg O([f_i(x)]) \vee [f_i(x)] \approx g_1(c)$		
[Resolving 16 with 14]:	(17) $\neg T(x) \vee \underline{\neg O([f_i(x)])} \vee [f_i(x)] \approx g_1(c) \vee [f_i(x)] \approx g_2(c)$		
[Resolving 10 with 17]:	**(18)** $\neg T(x) \vee \boldsymbol{\neg \overline{U_i(x)} \vee \neg O(x)} \vee \underline{\underline{[f_i(x)]}} \approx g_1(c) \vee [f_i(x)] \approx g_2(c)$		
The result after simplifying (18) with (11):			
(19)	$\neg T(x) \vee \neg U_i(x) \vee \neg O(x) \vee g_1(c) \approx c \vee g_2(c) \approx c$		

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 express (18) equivalently using an additional closure (19). It is easy to see that (19) is a logical consequence of (11) and (18). Furthermore, (19) makes (18) redundant, since (18) follows from smaller closures (11) and (19). Thus, (18) can be deleted from the closure set, which eventually ensures termination of the saturation.

4.2 The Saturation Strategy for Deciding Satisfiability of $\Gamma(KB)$

We say that N is a set of *DL-closures* if every closure in N is of some form from Table 5. Unary predicate symbols in N are organized into two sets $\mathcal{A}$ and $\mathcal{B}$.

Lemma 2. *For KB a $\mathcal{SHOIQ}$ knowledge base, $\Gamma(KB)$ is a set of DL-closures.*

Proof. The set $\Xi(KB)$ contains only closures from Table 3, and, due to Definition 1, each closure in $\Gamma(KB)$ contains a guard literal for each variable. □

To obtain a procedure for checking satisfiability of $\Gamma(KB)$, we next 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. 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.

Table 5. 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 $(*)$: 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: $(\neg) R(c,d)$ $(\neg) R(c, f(d))$ $(\neg) R(f(c), d)$ $\square$

$\mathcal{A} \subseteq \mathcal{B}$ are sets of predicate symbols; $\mathcal{A}$ contains all predicate symbols of $\Gamma(KB)$.
Each variable in each closure is guarded (see Definition 1).
$\alpha(x)$ 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.
c and d are constants, and $t_{(i)}$ are ground terms of the form c or $f(c)$.

Definition 2. *With $\mathcal{BS}_{DL}$ we denote the $\mathcal{BS}$ calculus parametrised 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 $R \in \mathcal{A}$, and unary predicates $A \in \mathcal{A}$ and $B \in \mathcal{B} \setminus \mathcal{A}$. 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 6, 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 this enables applying a simplification rule.

An example of an ordering suitable for $\mathcal{BS}_{DL}$ is a Knuth-Bendix ordering (KBO) [3] with $\mathsf{weight}(f) > \mathsf{weight}(R) > \mathsf{weight}(A) > \mathsf{weight}(B) > \mathsf{weight}(c) > \mathsf{weight}(\mathsf{tt})$, for each 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 symbol c.

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

Lemma 3 (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}:

Table 6. Simplification Rules of $\mathcal{BS}_{DL}$

Decomposition 1: $\dfrac{D \cdot \rho \vee L \cdot \rho}{D \cdot \rho \vee A(x),\ \neg A(x) \vee L \cdot \rho}$	(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 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: $\dfrac{D \cdot \rho \vee f(x) \approx c}{D \cdot \rho \vee B(x),\ \neg B(x) \vee f(x) \approx c}$	(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 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: $\dfrac{\alpha(x) \vee \bigvee_{i=1}^{n} [f(x)] \approx t_i}{\alpha(x) \vee \bigvee_{i=1}^{k} f(x) \approx c_i,\ \alpha(x) \vee \bigvee_{j=1}^{n} c_i \approx t_j \quad (1 \leq i \leq k)}$	(i) Some t_i is of the form $h(c)$; (ii) If Nominal Generation 1 has already been applied to some $\alpha(x) \vee \bigvee_{i=1}^{n_1} [f(x)] \approx t'_i$ (with 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: $\dfrac{\alpha(x) \vee \bigvee_{i=1}^{n} [f(x)] \approx t_i \vee \bigvee_{i=1}^{m} x \approx c_i}{\alpha(x) \vee \bigvee_{i=1}^{k} B_i(x),\ \neg B_i(x) \vee f(x) \approx e_i,\ \neg B_i(x) \vee x \approx d_i,\ \alpha(x) \vee \bigvee_{j=1}^{n} e_i \approx t_j \vee \bigvee_{j=1}^{m} d_i \approx c_j \quad (1 \leq i \leq k)}$	(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 been applied to some $\alpha(x) \vee \bigvee_{i=1}^{n_1} [f(x)] \approx t'_i \vee \bigvee_{i=1}^{m_1} x \approx c'_i$ (with 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.

(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 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 $x \approx g(f(x))$. 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 can only be 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 $o_1, \ldots, o_\ell$, $1 \leq \ell \leq n+m$. 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, e_i as $f(o_1)$, and make B_i true only for o_1.

Hence, if $\alpha(x)$ is true for all x, all conclusions are obviously true in I_s. Otherwise, for every x such that $\alpha(x)$ 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 4 (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 6, 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 2, 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 Condition (i), $D\rho$ contains either $h(x)$, but then $h(u) \succ B(u)$, or $D\rho$ contains $(\neg)A(x)$, but then $(\neg)A(u) \succ B(u)$ by Definition 2. Hence, $D\rho\tau \succ B(u)$, so the rest 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 $D = \alpha(u) \vee \bigvee_{i=1}^{k} f(u) \approx c_i$ into $E_i = \alpha(u) \vee \bigvee_{j=1}^{n} c_i \approx t_j$. 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 as follows: 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 $C \succ F$. □

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

We now show that $\mathcal{BS}_{DL}$ inferences on DL-closures always produce a DL-closure.

Table 7. 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	$\boxed{(\neg)R(f(c),d)}$		

Lemma 5 (Preservation of DL-Closures). *Let N be a set of DL-closures to which no $\mathcal{BS}_{DL}$ simplification 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 since 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 (that is not also of type 5) can contain a literal $x \not\approx c$, which is then selected. Otherwise, the closure must contain a literal $(\neg)B([f(x)])$: if this were not the case, the closure would have the form $\alpha(x) \vee \bigvee [f(x)] \approx t_i \vee \bigvee x \approx c_i$; 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 2, a literal of this form is eligible for inferences. The cases for the remaining closures are straightforward and are summarized in Table 7.

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 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, either into a term $f(x)$ of 1, 3, 4, or 9, a term $f([g(x)])$ of 2, or 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 other premise remains of the same type or becomes ground. 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 the 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, which is deleted, or a closure with a literal $x \not\approx x$, which is removed by reflexivity resolution and subsumption deletion.

Superposition from 4 into 1, 2, 3, 4, or 9 results in these 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 it 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 is either 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 6, and for the remaining standard ones this is trivial. □

4.4 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 $|KB|$. To the best of our knowledge, this assumption is used in all practical DL reasoning algorithms.

Lemma 6. *Let $\Gamma(KB) = N_0, N_1, \ldots, N_n$ be a $\mathcal{BS}_{DL}$ saturation. 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 predicates from $\Gamma(KB)$, $\alpha(x)$ can contain the predicates 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 total number of constants can increase 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 B for a combination of f and c, and Nominal Generation 2 introduces at most one predicate B_i for each e_i or d_i. Hence, the number of predicates 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 1. *$\mathcal{BS}_{DL}$ 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 2 and 5, each set of closures in a $\mathcal{BS}_{DL}$ saturation contains only DL-closures, and is at most triply exponential in size by Lemma 6. 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 3 and 4, the set of closures upon termination is saturated up to redundancy. Hence, $\Gamma(KB)$, and by Lemma 1 KB as well, is 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. □

5 Discussion and 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. The high complexity of our procedure is due to a possibly doubly exponential number of constants introduced by Nominal Generation 1 and 2. To understand the situations in which this can happen, consider the following example.

Let KB be a knowledge base from Table 8, which uses the well-known encoding of binary numbers by DL concepts. A concept B_i represents the i-th bit of a number. Thus, a number $b_p\, b_{p-1} \dots b_0$ with $b_i \in \{0, 1\}$ is represented by a concept $\mu_p(b_p) \sqcap \dots \sqcap \mu_0(b_0)$, where $\mu_i(0) = \neg B_i$ and $\mu_i(1) = B_i$. Axioms T1 and T2

Table 8. Expressing Big Cardinality Restrictions in $\mathcal{SHOIQ}$

T1.	$\top \sqsubseteq \leqslant 2\, R.\top$	T2.	$\top \sqsubseteq \geqslant 1\, R^{-}.\top$
T3.	$B_0 \sqsubseteq \forall R.\neg B_0$	T4.	$B_{i+1} \sqcap B_i \sqsubseteq \forall R.((B_{i+1} \sqcap B_i) \sqcup (\neg B_{i+1} \sqcap \neg B_i))$
T5.	$\neg B_0 \sqsubseteq \forall R.B_0$	T6.	$\neg B_{i+1} \sqcap B_i \sqsubseteq \forall R.((\neg B_{i+1} \sqcap B_i) \sqcup (B_{i+1} \sqcap \neg B_i))$
T7.	$\neg B_p \sqcap \cdots \sqcap \neg B_0 \sqsubseteq \{c\}$		$i = 1 \ldots p$
T8.	$\top \sqsubseteq \geqslant 2\, S.\top$	T9.	$\top \sqsubseteq \leqslant 1\, S^{-}.\top$
T10.	$A_0 \sqsubseteq \forall S.\neg A_0$	T11.	$A_{i+1} \sqcap A_i \sqsubseteq \forall S.((A_{i+1} \sqcap A_i) \sqcup (\neg A_{i+1} \sqcap \neg A_i))$
T12.	$\neg A_0 \sqsubseteq \forall S.A_0$	T13.	$\neg A_{i+1} \sqcap A_i \sqsubseteq \forall S.((\neg A_{i+1} \sqcap A_i) \sqcup (A_{i+1} \sqcap \neg A_i))$
A1.	$\neg A_q \sqcap \cdots \sqcap \neg A_0(c)$		$i = 1 \ldots q$
	T1—T7 express $\lvert B_p \sqcap \cdots \sqcap B_0 \rvert \leq 2^{2^p}$; T8—T12, A1 express $\lvert A_q \sqcap \cdots \sqcap A_0 \rvert \geq 2^{2^q}$		

ensure that a model of KB can be embedded into a binary R-tree: every element has at most two R-successors and at least one R-predecessor. Axioms T3–T6 ensure that the numbers $b_p\, b_{p-1} \ldots b_0$ corresponding to elements connected by R-links are incremented by one. Together with axiom T7, this ensures that the number of elements in the concept at the k-th level of this tree is at most 2^k. In particular, the last level, corresponding to the concept $B_p \sqcap \cdots \sqcap B_0$, can contain at most 2^{2^p} elements. Using a dual set of axioms T8–T13 and A1, we express in a similar way that the concept $A_q \sqcap \cdots \sqcap A_0$ contains at least 2^{2^q} objects.

Now checking subsumption between $A_q \sqcap \cdots \sqcap A_0$ and $B_p \sqcap \cdots \sqcap B_0$ w.r.t. KB amounts to testing whether a set with 2^{2^q} elements can be embedded into another set with 2^{2^p} elements. Such combinatorial problems, commonly called the *pigeon hole principle*, are known to be very hard for resolution [9]. On KB, our algorithm applies the Nominal Generation rules for all possible $\alpha(x) = (\neg)B_p(x) \vee \cdots \vee (\neg)B_0(x)$ and introduces a doubly exponential number of constants, because the constraint $\lvert B_p \sqcap \cdots \sqcap B_0 \rvert \leq 2^{2^p}$ is represented using a fresh constant for each element of the set. Although this observation does not prove that an optimal resolution-based procedure for $\mathcal{SHOIQ}$ cannot exist, it suggests that resolution alone may not suffice. In our future work, we shall investigate if it is possible to integrate algebraic reasoning directly into resolution—for example, as this was done for tableau calculi in [8].

However, worst-case complexity does not say anything about the typical case. Namely, the previous example causes problems because it succinctly encodes binary numbers. However, many applications do not require much combinatorial reasoning, so, on them, our algorithm does not introduce too many new constants. In fact, the Nominal Generation rules are triggered by terms $g(c)$, which can only result from interaction between inverse roles, number restrictions, and nominals. If these constructs are not used simultaneously, our algorithm becomes similar to the algorithm for the DL $\mathcal{SHIQ}$ presented in [11], and it runs in exponential time. Thus, our algorithm exhibits "pay-as-you-go" behavior. We shall implement our new algorithm in KAON2 to see how it behaves in practice.

References

1. A. Armando, S. Ranise, and M. Rusinowitch. Uniform Derivation of Decision Procedures by Superposition. In *Proc. CSL' 01*, volume 2142 of *LNCS*, pages 549–563, Paris, France, September 10–13 2001. Springer.

2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.
3. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
4. L. Bachmair and H. Ganzinger. Resolution Theorem Proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.
5. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation. *Information and Computation*, 121(2):172–192, 1995.
6. C. Fermüller, T. Tammet, N. Zamov, and A. Leitsch. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI*. Springer, 1993.
7. H. Ganzinger and H. de Nivelle. A Superposition Decision Procedure for the Guarded Fragment with Equality. In *Proc. LICS '99*, pages 295–305, Trento, Italy, July 2–5 1999. IEEE Computer Society.
8. V. Haarslev, M. Timmann, and R. Möller. Combining Tableaux and Algebraic Methods for Reasoning with Qualified Number Restrictions. In *Proc. DL 2001*, volume 49 of *CEUR Workshop Proceedings*, Stanford, CA, USA, August 1–3 2001.
9. A. Haken. The Intractability of Resolution. *Theorerical Computer Science*, 39:297–308, 1985.
10. I. Horrocks and U. Sattler. A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. IJCAI 2005*, pages 448–453, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.
11. U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In *Proc. KR 2004*, pages 152–162, Whistler, Canada, June 2–5 2004. AAAI Press.
12. W. H. Joyner Jr. Resolution Strategies as Decision Procedures. *Journal of the ACM*, 23(3):398–417, 1976.
13. Y. Kazakov and H. de Nivelle. A Resolution Decision Procedure for the Guarded Fragment with Transitive Guards. In *Proc. IJCAR 2004*, volume 3097 of *LNAI*, pages 122–136, Cork, Ireland, July 4–8 2004. Springer.
14. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.
15. R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19(4):312–351, 1995.
16. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-Based Methods for Modal Logics. *Logic Journal of the IGPL*, 8(3):265–292, 2000.
17. A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.
18. L. Pacholski, W. Szwast, and L. Tendera. Complexity Results for First-Order Two-Variable Logic with Counting. *SIAM Journal on Computing*, 29(4):1083–1117, 2000.
19. I. Pratt-Hartmann. Complexity of the Two-Variable Fragment with Counting Quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.
20. R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In *Proc. CADE-19*, volume 2741 of *LNAI*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003. Springer.
21. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.