Optimized Reasoning in Description Logics using Hypertableaux

Boris Motik, Rob Shearer, and Ian Horrocks

University of Manchester, UK

Abstract. We present a novel reasoning calculus for Description Logics (DLs)—knowledge representation formalisms with applications in areas such as the Semantic Web. In order to reduce the nondeterminism due to general inclusion axioms, we base our calculus on hypertableau and hyperresolution calculi, which we extend with a blocking condition to ensure termination. To prevent the calculus from generating large models, we introduce *"anywhere" pairwise blocking*. Our preliminary implementation shows significant performance improvements on several well-known ontologies. To the best of our knowledge, our reasoner is currently the only one that can classify the original version of the GALEN terminology.

1 Introduction

Description Logics (DLs) [2]—knowledge representation formalisms with well-understood formal properties—have been applied to numerous problems in computer science. A central component of most DL applications is an efficient and scalable reasoner. Modern reasoners, such as Pellet [15], FaCT++ [21], and RACER [8], are typically based on tableau calculi [2, Chapter 2]. These calculi demonstrate (un)satisfiability of a knowledge base $\mathcal{K}$ via a constructive search for an abstraction of a model of $\mathcal{K}$. Numerous optimizations have been developed in an effort to reduce the size of the search space [2, Chapter 9].

Despite major advances in recent years, ontologies are still encountered in practice that cannot be handled by existing reasoners. This is mainly because many different models might need to be examined, and each model might be very large [2, Chapter 3]. The former problem is due to *or-branching*: given a disjunctive assertion $C \sqcup D(s)$, a tableau algorithm nondeterministically guesses that either $C(s)$ or $D(s)$ holds. To show unsatisfiability of $\mathcal{K}$, *every* possible guess must lead to a contradiction: if assuming $C(s)$ leads to a contradiction, the algorithm must backtrack and assume $D(s)$. This can clearly result in exponential behavior. General concept inclusions (GCIs)—axioms of the form $C \sqsubseteq D$—are the main source of disjunctions: to ensure that $C \sqsubseteq D$ holds, a tableau algorithm adds a disjunction $\neg C \sqcup D(s)$ to each individual s in the model. Various *absorption* optimizations [2, Chapter 9][11, 20] reduce the high degree of nondeterminism in such a procedure; however, they often fail to eliminate all sources of nondeterminism. This may be the case even for ontologies that can be translated into Horn clauses (such as GALEN, NCI, and SNOMED), for which reasoning without any nondeterminism should be possible in principle.

The size of the model being constructed is determined by *and-branching*—the expansion of a model due to existential quantifiers. Apart from memory consumption problems, and-branching can increase or-branching by increasing the number of individuals to which GCIs are applied.

In this paper, we present a reasoning calculus that addresses both sources of complexity. We focus on the DL $\mathcal{SHIQ}$; however, our calculus should be applicable to most DLs with known tableau algorithms. A $\mathcal{SHIQ}$ knowledge base is first preprocessed into *DL-clauses*—universally quantified implications containing DL concepts and roles as predicates. The main inference rule for DL-clauses is hyperresolution: an atom from the head of a DL clause is derived *only if* all atoms from the clause body have been derived. On Horn clauses, this calculus is deterministic, which eliminates all or-branching. This is in contrast with existing DL tableau calculi, which often behave nondeterministically on Horn problems. Our algorithm can thus be viewed as a hybrid of resolution and tableau, and is related to the hypertableau [3] and hyperresolution [17] calculi.

Hyperresolution decides many first-order fragments (see, e.g., [6, 5] for an overview). Unlike most of these fragments, $\mathcal{SHIQ}$ allows for cyclic GCIs of the form $C \sqsubseteq \exists R.C$, on which hyperresolution can generate infinite paths of successors. Therefore, to ensure termination, we use the pairwise blocking technique from [10] to detect cyclic computations. Due to hyper-inferences, the soundness and correctness proofs from [10] do not carry over to our calculus. In fact, certain simpler blocking conditions for weaker DLs cannot be applied in a straightforward manner in our setting. To limit and-branching, we extend the blocking condition from [10] to *anywhere pairwise blocking*: an individual can be blocked by an individual that is not necessarily an ancestor. This significantly reduces the sizes of the constructed models. Anywhere blocking has already been used with *subset* blocking [1]; however, to the best of our knowledge, it has neither been used with the more sophisticated pairwise blocking nor tested in practice.

We have implemented our calculus in a new reasoner. Even with a relatively naïve implementation, our reasoner outperforms existing reasoners on several real-world ontologies. For example, the deterministic treatment of GCIs significantly reduces the classification time for the NCI ontology. Furthermore, the pairwise anywhere blocking strategy seems to be very effective in limiting model sizes. To the best of our knowledge, our reasoner is currently the only one that can classify the original version of the GALEN terminology.

2 Preliminaries

The DL $\mathcal{SHIQ}$ is defined as follows. For N_R a set of *atomic roles*, the set of *roles* is $N_R \cup \{R^- \mid R \in N_R\}$. For $R \in N_R$, let $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. An *RBox* $\mathcal{R}$ 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 $\{R \sqsubseteq S, \mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\}$. A role R is *transitive* in $\mathcal{R}$ if a role S exists such that $S \sqsubseteq^* R$, $R \sqsubseteq^* S$, and either $\mathsf{Trans}(S) \in \mathcal{R}$ or $\mathsf{Trans}(\mathsf{Inv}(S)) \in \mathcal{R}$; R is *simple* if no transitive role S exists with $S \sqsubseteq^* R$.

Table 1. Model-Theoretic Semantics of $\mathcal{SHIQ}$

Semantics of Roles and Concepts	Semantics of Axioms
$\top^I = \triangle^I \qquad \bot^I = \emptyset$	$C \sqsubseteq D \quad \Rightarrow C^I \subseteq D^I$
$(\neg C)^I = \triangle^I \setminus C^I \qquad (R^-)^I = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^I\}$	$R \sqsubseteq S \quad \Rightarrow R^I \subseteq S^I$
$(C \sqcap D)^I = C^I \cap D^I \quad (C \sqcup D)^I = C^I \cup D^I$	$\mathsf{Trans}(R) \Rightarrow (R^I)^+ \subseteq R^I$
$(\forall R.C)^I = \{x \mid \forall y : \langle x, y\rangle \in R^I \rightarrow y \in C^I\}$	$C(a) \quad \Rightarrow a^I \in C^I$
$(\exists R.C)^I = \{x \mid \exists y : \langle x, y\rangle \in R^I \wedge y \in C^I\}$	$R(a, b) \quad \Rightarrow \langle a^I, b^I\rangle \in R^I$
$(\leq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \leq n\}$	$a \approx b \quad \Rightarrow a^I = b^I$
$(\geq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x, y\rangle \in S^I \wedge y \in C^I\} \geq n\}$	$a \not\approx b \quad \Rightarrow a^I \neq b^I$

Note: $\sharp N$ is the number of elements in N, and R^+ is the transitive closure of R.

For a set of *atomic concepts* N_C, the set of *concepts* is the smallest set containing $\top$, $\bot$, A, $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\geq n\, S.C$, and $\leq n\, S.C$, for $A \in N_C$, C and D concepts, R a role, S a simple role, and n a nonnegative integer. A TBox $\mathcal{T}$ is a finite set of *general concept inclusions* (GCIs) $C \sqsubseteq D$.

For a set of *individuals* N_I, an ABox $\mathcal{A}$ is a finite set of assertions $C(a)$, $R(a, b)$, and (in)equalities $a \approx b$ and $a \not\approx b$, where C is a concept, R is a role, and a and b are individuals. A $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$.

An *interpretation* for $\mathcal{K}$ is a tuple $I = (\triangle^I, \cdot^I)$, where $\triangle^I$ is a nonempty set, and $\cdot^I$ assigns an element $a^I \in \triangle^I$ to each individual a, a set $A^I \subseteq \triangle^I$ to each atomic concept A, and a relation $R^I \subseteq \triangle^I \times \triangle^I$ to each atomic role R. The function $\cdot^I$ is extended to concepts and roles as shown in the left-hand side of Table 1. I is a *model* of $\mathcal{K}$, written $I \models \mathcal{K}$, if it satisfies all axioms of $\mathcal{K}$ as shown in the right-hand side of Table 1. The basic inference problem for $\mathcal{SHIQ}$ is checking satisfiability of $\mathcal{K}$—that is, checking whether a model of $\mathcal{K}$ exists.

The *negation-normal form* of a concept C, written $\mathsf{nnf}(C)$, is the concept equivalent to C containing negations only in front of atomic concepts; $\dot{\neg} C$ is an abbreviation for $\mathsf{nnf}(\neg C)$. $|\mathcal{K}|$ is the size of $\mathcal{K}$ with numbers coded in unary. The DL $\mathcal{ALCHIQ}$ is obtained from $\mathcal{SHIQ}$ by disallowing transitive roles.

3 Algorithm Overview

To see how GCIs can increase or-branching and thus cause performance problems, consider the following knowledge base $\mathcal{K}_1$:

$$(1) \quad \begin{array}{l} \mathcal{T}_1 = \{\exists R.A \sqsubseteq A\} \\ \mathcal{A}_1 = \{\neg A(a_0),\ R(a_0, b_1),\ R(b_1, a_1), \ldots,\ R(a_{n-1}, b_n),\ R(b_n, a_n),\ A(a_n)\} \end{array}$$

To satisfy the GCI, a tableau algorithm derives $(\forall R.\neg A \sqcup A)(a_i)$, $0 \leq i \leq n$ and $(\forall R.\neg A \sqcup A)(b_j)$, $1 \leq j \leq n$. Assuming that a_i are processed before b_j, the algorithm derives $\forall R.\neg A(a_i)$, $0 \leq i \leq n$ and $\neg A(b_i)$, $1 \leq i \leq n$, after which it derives $\forall R.\neg A(b_i)$, $1 \leq i \leq n-1$ and $\neg A(a_i)$, $1 \leq i \leq n$. The ABox now contains a contradiction on a_n, so the algorithm flips its guess on b_{n-1} to $A(b_{n-1})$. This generates a contradiction on b_{n-1}, so the algorithm backtracks from all guesses for b_i. Next, the guess on a_n is changed to $A(a_n)$ and the work for all b_i is repeated. This also leads to a contradiction, so the algorithm must revise its guess

for a_{n-1}; but then, two guesses are again possible for a_n. In general, after revising a guess for a_i, all possibilities for a_j, $i < j \leq n$, must be reexamined, which results in exponential behavior. Note that none of the standard backtracking optimizations [2, Chapter 9] help us avoid this problem. Namely, the problem arises because the order in which the individuals are processed makes the guesses on a_i independent from the guesses on a_j, $i \neq j$. It is difficult to estimate in advance which order is optimal; in fact, the processing order is typically determined by implementation side-effects (such as the data structures used to store $\mathcal{K}$).

The GCI $\exists R.A \sqsubseteq A$ is not inherently nondeterministic: it is equivalent to the Horn clause $\forall x, y : [R(x, y) \wedge A(y) \rightarrow A(x)]$. By hyperresolution, we derive the facts $A(b_n), A(a_{n-1}), \ldots, A(a_0)$, and eventually we derive a contradiction on a_0. These inferences are deterministic, so we can conclude that $\mathcal{K}_1$ is unsatisfiable without any backtracking. This example suggests that the way tableau algorithms handle GCIs can be "unnecessarily" nondeterministic.

Absorption [2, Chapter 9] reduces the nondeterminism introduced by GCIs. If possible, it rewrites GCIs as $B \sqsubseteq C$ with B an atomic concept; then, during reasoning, it derives $C(s)$ only if the ABox contains $B(s)$. This localizes the applicability of the rewritten GCIs. Absorption has been extended to *binary absorption* [11], which rewrites a GCI to $B_1 \sqcap B_2 \sqsubseteq C$, and to *role absorption* [20], which rewrites a GCI to $\exists R.\top \sqsubseteq C$. Note, however, that the axiom $\exists R.A \sqsubseteq A$ cannot be absorbed directly. It can be absorbed if it is rewritten as $A \sqsubseteq \forall R^-.A$. In practice, it is often unclear in advance which combination of transformation and absorption techniques will yield the best results. Therefore, implemented absorption algorithms are guided primarily by heuristics.

Our algorithm can be seen as a generalization of absorption. It first translates GCIs into *DL-clauses*—universally quantified implications of the form $\bigwedge U_i \rightarrow \bigvee V_j$, where U_i are of the form $R(x, y)$ or $A(x)$, and V_j are of the form $R(x, y)$, $A(x)$, $\exists R.C(x)$, $\geq n\, R.C(x)$, or $x \approx y$. DL-clauses are used in *hyperresolution* inferences, which derive some V_j, but only if all U_i are matched to assertions in the ABox. This calculus is quite different from the standard DL tableau calculi. For example, it has no *choose*-rule for qualified number restrictions [19], and it can handle implications such as $R(x, y) \rightarrow B(x) \vee A(y)$ (obtained from $\exists R.\neg A \sqsubseteq B$) that contain several universally quantified variables.

It is easy to see that and-branching can cause the introduction of infinitely many new individuals. Consider the following (satisfiable) knowledge base:

$$(2)\quad \mathcal{T}_2 = \left\{ \begin{array}{l} A_1 \sqsubseteq \geq 2\, S.A_2,\ \ldots,\ A_{n-1} \sqsubseteq \geq 2\, S.A_n,\ A_n \sqsubseteq A_1, \\ A_i \sqsubseteq (B_1 \sqcup C_1) \sqcap \ldots \sqcap (B_m \sqcup C_m) \text{ for } 1 \leq i \leq n \end{array} \right\}\ \mathcal{A}_2 = \{A_1(a)\}$$

To check satisfiability of $\mathcal{K}_2$, a tableau algorithm builds a binary tree with each node labeled with some A_i and an element of $\Pi = \{B_1, C_1\} \times \ldots \times \{B_m, C_m\}$. A naïve algorithm would try to construct an infinite tree, so tableau algorithms employ *blocking* [10]: if a node a is labeled with the same concepts as some ancestor a' of a, then the existential quantifiers for a are not expanded. This ensures termination; however, the number of elements in Π is exponential, so, with "unlucky" guesses, the tree can be exponential in depth and doubly exponential

in total. In the best case, the algorithm can, for example, choose B_j rather than C_j for each $1 \leq j \leq m$. It then constructs a polynomially deep binary tree and thus runs in exponential time.

To curb and-branching, we extend pairwise blocking [10] to *anywhere pairwise blocking*, in which an individual can be blocked not only by an ancestor, but by any individual satisfying certain ordering requirements. This reduces the worst-case complexity of the tableau algorithm by an exponential factor; for example, on $\mathcal{K}_2$, after we exhaust all members of Π, all subsequently created individuals will be blocked. Such blocking can sometimes also improve the best-case complexity; for example, on $\mathcal{K}_2$, our algorithm can create a polynomial path and then use the individuals from that path to block their siblings.

We explain the remaining two aspects of our algorithm. First, the $\forall_+$-rule traditionally used to deal with transitive roles does not work in our setting, since we represent concepts of the form $\forall R.C$ as DL-clauses. Therefore, we encode transitivity axioms using GCIs in a preprocessing step. Second, to avoid an exponential blowup in the transformation of GCIs to DL-clauses, we apply the well-known *structural transformation* [16]. We take special care, however, to maximize the likelihood that the result can be translated into Horn DL-clauses. For example, given $\top \sqsubseteq \forall R.(C \sqcup \forall S.\neg D)$, if we replace $\forall S.\neg D$ with an atomic concept Q, we obtain the axioms (3), of which the first one does not give us a Horn DL-clause. If, however, we replace $\forall S.\neg D$ with $\neg Q'$, we obtain the axioms (4), which can both be translated into Horn DL-clauses.

$$(3) \qquad \begin{array}{lcl} \top \sqsubseteq \forall R.(C \sqcup Q) & \rightsquigarrow & R(x,y) \rightarrow C(y) \vee Q(y) \\ Q \sqsubseteq \forall S.\neg D & \rightsquigarrow & Q(x) \wedge S(x,y) \wedge D(y) \rightarrow \bot \end{array}$$

$$(4) \qquad \begin{array}{lcl} \top \sqsubseteq \forall R.(C \sqcup \neg Q') & \rightsquigarrow & R(x,y) \wedge Q'(y) \rightarrow C(y) \\ \neg Q' \sqsubseteq \forall S.\neg D & \rightsquigarrow & S(x,y) \wedge D(y) \rightarrow Q'(x) \end{array}$$

In Section 4.1, we present a version of the structural transformation that replaces a complex concept with either a positive or negative atomic concept, depending on the polarity of the concept being replaced. We thus bring GCIs into a *normalized* form in which no complex concept occurs under implicit negations; then, we translate such GCIs into DL-clauses.

4 The Satisfiability Checking Algorithm

4.1 Preprocessing

Elimination of Transitivity Axioms. We first encode a $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base $\Omega(\mathcal{K})$. Roughly speaking, an 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 a "relevant" concept from $\mathcal{K}$. This encoding is polynomial and has been presented several times for various description [19] and modal [18] logics. Therefore, we omit the details of the transformation and refer the reader to [14, Section 5.2]. After this transformation, there is no distinction between simple and complex roles, so, without loss of generality, in the rest of this paper we treat $\exists R.C$ as a syntactic shortcut for $\geq 1\, R.C$.

Table 2. The Functions Used in the Structural Transformation

$$\begin{aligned}
\Delta(\mathcal{K}) &= \{\top(\iota)\} \cup \textstyle\bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \Delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \Delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2)) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup C') &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \textstyle\bigcup_{i=1}^{n} \Delta(\alpha_{C'} \sqsubseteq C_i) \text{ for } C' = \prod_{i=1}^{n} C_i \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.\alpha_D) \cup \Delta(\alpha_D \sqsubseteq D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.\alpha_D) \cup \Delta(\alpha_D \sqsubseteq D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.\dot{\neg}\alpha_{D'}) \cup \Delta(\alpha_{D'} \sqsubseteq D') \text{ for } D' = \dot{\neg} D \\
\Delta(D(a)) &= \{\alpha_D(a)\} \cup \Delta(\alpha_D \sqsubseteq D) \\
\Delta(R^{-}(a,b)) &= \{R(b,a)\} \\
\Delta(\beta) &= \{\beta\} \text{ for any other axiom } \beta
\end{aligned}$$

$$\alpha_C = \begin{cases} Q_C & \text{if } \mathsf{pos}(C) = \mathsf{true} \\ \neg Q_C & \text{if } \mathsf{pos}(C) = \mathsf{false} \end{cases} \quad \text{where } Q_C \text{ is a fresh atomic concept unique for } C$$

$$\begin{aligned}
\mathsf{pos}(\top) &= \mathsf{false} & \mathsf{pos}(\bot) &= \mathsf{false} \\
\mathsf{pos}(A) &= \mathsf{true} & \mathsf{pos}(\neg A) &= \mathsf{false} \\
\mathsf{pos}(C_1 \sqcap C_2) &= \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) & \mathsf{pos}(C_1 \sqcup C_2) &= \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) \\
\mathsf{pos}(\forall R.C_1) &= \mathsf{pos}(C_1) & \mathsf{pos}(\leq n\, R.C_1) &= \begin{cases} \mathsf{pos}(\dot{\neg} C_1) & \text{if } n = 0 \\ \mathsf{true} & \text{otherwise} \end{cases} \\
\mathsf{pos}(\geq n\, R.C_1) &= \mathsf{true}
\end{aligned}$$

Note: A is an atomic concept, C_i are arbitrary concepts, $\mathbf{C}$ is a possibly empty disjunction of arbitrary concepts, D is not a literal concept, and ι is a fresh individual.

Structural Transformation. GCIs are next brought into a certain normalized form, defined as follows:

Definition 1. *For A an atomic concept, the concepts A, $\neg A$, $\top$, and $\bot$ are called* literal concepts. *A GCI is* normalized *if it is of the form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, where each C_i is of the form B, $\forall R.B$, $\geq n\, R.B$, or $\leq n\, R.B$, and B is a literal concept. A TBox $\mathcal{T}$ is* normalized *if all GCIs in it are normalized. An ABox $\mathcal{A}$ is* normalized *if (*i*) each concept assertion in $\mathcal{A}$ is of the form $B(s)$ or $\geq n\, R.B(s)$, for B a literal concept, (*ii*) each role assertion in $\mathcal{A}$ contains only atomic roles, and (*iii*) $\mathcal{A}$ contains at least one assertion. A knowledge base $\mathcal{K}$ is* normalized *if $\mathcal{T}$ and $\mathcal{A}$ are normalized.*

A knowledge base $\mathcal{K}$ can be brought into normalized form $\Delta(\mathcal{K})$ as follows:

Definition 2. *For $\mathcal{K}$ an $\mathcal{ALCHIQ}$ knowledge base, $\Delta(\mathcal{K})$ is the knowledge base computed as shown in Table 2.*

The difference between the well-known structural transformation [16] and Definition 2 is as follows. Assume that we need to rename a nonatomic subconcept D of C. If $\mathsf{pos}(D) = \mathsf{false}$, then D can be converted into clauses with only negative literals, so we rename D by a negative literal concept $\neg Q_D$; otherwise, the clausification of D requires at leat one positive literal, so we rename D by a positive literal concept Q_D. In this way, the renaming of D in C does not change the number of positive literals in the clausal representation of C, so renaming preserves Horn-ness. Furthermore, for a Horn-$\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ [12], the

knowledge base $\Delta(\mathcal{K})$ is guaranteed also to be a Horn-$\mathcal{SHIQ}$ knowledge base that can be translated into Horn DL-clauses.

Lemma 1. *An $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K}$ is satisfiable if and only if $\Delta(\mathcal{K})$ is satisfiable; $\Delta(\mathcal{K})$ can be computed in polynomial time; and $\Delta(\mathcal{K})$ is normalized.*

Proof. It is easy to see that our transformation is a syntactic variant of the structural transformation from [16], from which the first two claims follow. Observe that Δ essentially rewrites each GCI into a form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$ and then keeps replacing nested subconcepts of C_i as long as the GCI is not normalized. Furthermore, it adds $\top(\iota)$ to the ABox so that it is not empty, and it replaces all inverse role assertions with equivalent assertions on the atomic roles. □

Translation into DL-Clauses. We next define the notion of DL-clauses:

Definition 3. *Let N_V be a set of variables disjoint from N_I. An* atom *is an expression of the form $C(s)$, $R(s,t)$, or $s \approx t$, for s and t individuals or variables, C a concept, and R a role. A* DL-clause *is an expression of the form*

$$U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n \tag{5}$$

where U_i and V_j are atoms, $m \geq 0$, and $n \geq 0$. The conjunction $U_1 \wedge \ldots \wedge U_m$ is called the antecedent*, and the disjunction $V_1 \vee \ldots \vee V_n$ is called the* consequent*.*

Let $I = (\triangle^I, \cdot^I)$ be an interpretation and $\mu : N_V \rightarrow \triangle^I$ a variable mapping. Let $a^{I,\mu} = a^I$ for an individual a and $x^{I,\mu} = \mu(x)$ for a variable x. Satisfaction of an atom, DL-clause, and a set of DL-clauses N in I and μ is defined as follows:

$I, \mu \models C(s)$	*if* $s^{I,\mu} \in C^I$
$I, \mu \models R(s,t)$	*if* $\langle s^{I,\mu}, t^{I,\mu} \rangle \in R^I$
$I, \mu \models s \approx t$	*if* $s^{I,\mu} = t^{I,\mu}$
$I, \mu \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_n$	*if* $I, \mu \models V_j$ *for some* $1 \leq j \leq n$ *whenever* $I, \mu \models U_i$ *for each* $1 \leq i \leq m$
$I \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_n$	*if* $I, \mu \models \bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_n$ *for all mappings* μ
$I \models N$	*if* $I \models r$ *for each DL-clause* $r \in N$

In the rest of this paper, we assume that each atom $s \approx t$ ($s \not\approx t$) also stands for the symmetric atom $t \approx s$ ($t \not\approx s$). Furthermore, we allow ABoxes to contain the assertion $\bot$, which is false in all interpretations. Finally, we denote the empty consequents of DL-clauses with $\bot$. We now show how to transform a normalized $\mathcal{ALCHIQ}$ knowledge base into a set of DL-clauses.

Definition 4. *For a normalized $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$, the set of DL-clauses $\Xi(\mathcal{K})$ is obtained as shown in Table 3.*

To simplify the *Hyp*-rule in Section 4.2, the role atoms in $\mathcal{A}$ and $\Xi(\mathcal{K})$ involve only atomic roles. Thus, the function ar from Table 3 is used to convert inverse role atoms $R^-(s,t)$ in $\Xi(\mathcal{K})$ into atomic role atoms $R(t,s)$. An inverse role can occur only in concepts of the form $\geq n\, R^-.C$, so the $\geq$-rule (defined in Section 4.2) also uses ar to generate atoms with atomic roles.

Table 3. Translation of Normalized GCIs to DL-Clauses

$$\Xi(\mathcal{K}) = \{\left[\bigwedge_{i=1}^{n} \mathsf{lhs}(C_i)\right] \rightarrow \left[\bigvee_{i=1}^{n} \mathsf{rhs}(C_i)\right] \mid \text{ for each } \top \sqsubseteq \bigsqcup_{i=1}^{n} C_i \text{ in } \mathcal{T}\} \cup$$
$$\{\mathsf{ar}(R,x,y) \rightarrow \mathsf{ar}(S,x,y) \mid \text{ for each } R \sqsubseteq S \text{ in } \mathcal{R}\}$$
$$\mathsf{ar}(R,s,t) = \begin{cases} R(s,t) \text{ if } R \text{ is an atomic role} \\ S(t,s) \text{ if } R \text{ is an inverse role and } R = S^- \end{cases}$$

Note: Whenever $\mathsf{lhs}(C_i)$ or $\mathsf{rhs}(C_i)$ is undefined, it is omitted in the DL-clause.

C	$\mathsf{lhs}(C)$	$\mathsf{rhs}(C)$
A		$A(x)$
$\neg A$	$A(x)$	
$\geq n\, R.A$		$\geq n\, R.A(x)$
$\geq n\, R.\neg A$		$\geq n\, R.\neg A(x)$
$\forall R.A$	$\mathsf{ar}(R,x,y_C)$	$A(y_C)$
$\forall R.\neg A$	$\mathsf{ar}(R,x,y_C) \wedge A(y_C)$	
$\leq n\, R.A$	$\bigwedge_{i=1}^{n+1}[\mathsf{ar}(R,x,y_C^i) \wedge A({y_C}^i)]$	$\bigvee_{i=1}^{n+1}\,\bigvee_{j=i+1}^{n+1} y_C^i \approx y_C^j$
$\leq n\, R.\neg A$	$\bigwedge_{i=1}^{n+1} \mathsf{ar}(R,x,y_C^i)$	$\bigvee_{i=1}^{n+1} A(y_C^i) \vee \bigvee_{i=1}^{n+1}\,\bigvee_{j=i+1}^{n+1} y_C^i \approx y_C^j$

Note: Each variable $y_C^{(i)}$ is unique for C (and i), and it is different from x.

Lemma 2. *Let $\mathcal{K}$ be a normalized $\mathcal{ALCHIQ}$ knowledge base. Then, $I \models \mathcal{K}$ if and only if $I \models \Xi(\mathcal{K})$ and $I \models \mathcal{A}$.*

Proof. The following equivalences between DLs and first-order logic are known:

$$\forall R.C(x) \equiv \forall y : \neg R(x,y) \vee C(y)$$
$$\leq n\, R.C(x) \equiv \forall y_1, \ldots, y_{n+1} : \bigvee_{i=1}^{n+1}[\neg R(x,y_i) \vee \neg C(y_i)] \vee \bigvee_{i=1}^{n+1}\,\bigvee_{j=i+1}^{n+1} y_i \approx y_j$$

Clearly, $\Xi(\mathcal{K})$ is obtained from normalized GCIs by expanding the concepts $\forall R.C$ and $\leq n\, R.C$ according to these equivalences, and then moving all negative atoms into the antecedent and all positive atoms into the consequent. □

4.2 The Hypertableau Calculus for DL-Clauses

We now present our hypertableau calculus for deciding satisfiability of $\mathcal{A} \cup \Xi(\mathcal{K})$.

Definition 5. ***Unnamed Individuals.*** *For a set of* named *individuals N_I, the set of* all individuals *N_X is inductively defined as $N_I \subseteq N_X$ and, if $x \in N_X$, then $x.i \in N_X$ for each integer i. The individuals in $N_X \setminus N_I$ are* unnamed. *An individual $x.i$ is a* successor *of x, and x is a* predecessor *of $x.i$;* descendant *and* ancestor *are the transitive closures of successor and predecessor, respectively.*

Pairwise Anywhere Blocking. *A concept is* blocking-relevant *if it is of the form A, $\geq n\, R.A$, or $\geq n\, R.\neg A$, for A an atomic concept. The* label of an individual *s and* of an individual pair *$\langle s,t \rangle$ in an ABox $\mathcal{A}$ are defined as follows:*

$$\mathcal{L}_{\mathcal{A}}(s) = \{C \mid C(s) \in \mathcal{A} \text{ and } C \text{ is a blocking-relevant concept}\}$$
$$\mathcal{L}_{\mathcal{A}}(s,t) = \{R \mid R(s,t) \in \mathcal{A}\}$$

Let $\prec$ be a strict ordering (i.e., a transitive and irreflexive relation) on N_X containing the ancestor relation—that is, if s' is an ancestor of s, then $s' \prec s$. By induction on $\prec$, we assign to each individual s in $\mathcal{A}$ a status as follows:

- *s is* directly blocked by *an individual s' iff both s and s' are unnamed, s' is not blocked, $s' \prec s$, $\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(s')$, $\mathcal{L}_\mathcal{A}(t) = \mathcal{L}_\mathcal{A}(t')$, $\mathcal{L}_\mathcal{A}(s,t) = \mathcal{L}_\mathcal{A}(s',t')$, and $\mathcal{L}_\mathcal{A}(t,s) = \mathcal{L}_\mathcal{A}(t',s')$, for t and t' the predecessors of s and s', resp.*
- *s is* indirectly blocked *iff its predecessor is blocked.*
- *s is* blocked *iff it is either directly or indirectly blocked.*

Pruning. *The ABox $\mathsf{prune}_\mathcal{A}(s)$ is obtained from $\mathcal{A}$ by removing all assertions of the form $R(t,t.i)$, $R(t.i,t)$, $C(t.i)$, $u \approx t.i$, and $u \not\approx t.i$, where t is either s or some descendant of s, i is an integer, and u is an arbitrary individual.*

Merging. *The ABox $\mathsf{merge}_\mathcal{A}(s \to t)$ is obtained from $\mathsf{prune}_\mathcal{A}(s)$ by replacing the individual s with the individual t in all assertions.*

Derivation Rules. *Table 4 specifies* derivation rules *that, given an ABox $\mathcal{A}$ and a set of DL-clauses $\Xi(\mathcal{K})$, derive the ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$. In the Hyp-rule, σ is a mapping from N_V to the individuals occurring in $\mathcal{A}$, and $\sigma(U)$ is the atom obtained from U by replacing each variable x with $\sigma(x)$.*

Derivation. *For a normalized $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$, a* derivation *is a pair (T, λ) where T is a finitely branching tree and λ is a function that labels the nodes of T with ABoxes such that (*i*) $\lambda(\epsilon) = \mathcal{A}$ for ϵ the root of the tree, and (*ii*) for each node t, if one or more derivation rules are applicable to $\lambda(t)$ and $\Xi(\mathcal{K})$, then t has children $t_1, \ldots, t_n$ such that $\lambda(t_1), \ldots, \lambda(t_n)$ are the result of applying one (arbitrarily chosen) applicable rule to $\lambda(t)$ and $\Xi(\mathcal{K})$.*

Clash. *An ABox $\mathcal{A}$ contains a* clash *iff $\bot \in \mathcal{A}$; otherwise, $\mathcal{A}$ is* clash-free.

In [10], the successor relation is encoded using role arcs, which point only from predecessors to successors. Since our ABoxes contain only atomic roles, role arcs can point in both directions, so we encode the successor relation in the individuals. The ordering $\prec$ ensures that there are no cyclic blocks, so all successors of nonblocked individuals have been constructed. *Ancestor pairwise blocking* from [10] is obtained if $\prec$ is exactly the descendant relation.

Pruning prevents infinite loops of merge-create rule applications—the so-called "yo-yo" effect. Consider the following example:

$$\begin{aligned} \mathcal{A}_3 &= \{A(a),\ \ \exists R.\top(a),\ \ R(a,b),\ \ R(a,a)\} \\ \Xi(\mathcal{K}_3) &= \{R(x,y_1) \wedge R(x,y_2) \to y_1 \approx y_2,\ \ A(x) \wedge R(x,y) \to \exists R.\top(y)\} \end{aligned} \tag{6}$$

By the second DL-clause, we derive $\exists R.\top(b)$, which we expand to $R(b, b.1)$. But then, by the first DL-clause, we derive $b \approx a$. Hence, we merge b into a, and obtain $\mathcal{A}_3' = \{A(a), \exists R.\top(a), R(a, b.1), R(a,a)\}$. The ABox $\mathcal{A}_3'$ is isomorphic to $\mathcal{A}_3$, so we can repeat the whole process, which clearly leads to nontermination. To remedy this, we remove all assertions that involve successors of b before merging b into a; we thus obtain $\mathcal{A}_3'' = \{A(a), R(a,a), \exists R.\top(a)\}$, after which the algorithm terminates. Intuitively, merging ensures that no individual "inherits" successors through merging. In [10], the successors are not physically removed,

Table 4. Derivation Rules of the Tableau Calculus

Hyp-rule	If 1. $U_1 \wedge ... \wedge U_m \rightarrow V_1 \vee ... \vee V_n \in \Xi(\mathcal{K})$, 2. a mapping $\sigma : N_V \rightarrow N_\mathcal{A}$ exists, for $N_\mathcal{A}$ the set of individuals in $\mathcal{A}$, 3. $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$, 4. $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$, then if $n = 0$, then $\mathcal{A}_1 = \mathcal{A} \cup \{\bot\}$, otherwise $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$.
$\geq$-rule	If 1. $\geq n\, R.C(s) \in \mathcal{A}$, 2. s is not blocked in $\mathcal{A}$, and 3. there are no individuals $u_1, \ldots, u_n$ such that $\{\mathsf{ar}(R, s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$, then $\mathcal{A}_1 := \mathcal{A} \cup \{\mathsf{ar}(R, s, t_i),\ C(t_i) \mid 1 \leq i \leq n\} \cup \{t_i \not\approx t_j \mid 1 \leq i < j \leq n\}$ where $t_1, \ldots, t_n$ are fresh pairwise distinct successors of s.
$\approx$-rule	If 1. $s \approx t \in \mathcal{A}$ and 2. $s \neq t$ then $\mathcal{A}_1 := \mathsf{merge}_\mathcal{A}(s \rightarrow t)$ if t is named or if s is a descendant of t, $\mathcal{A}_1 := \mathsf{merge}_\mathcal{A}(t \rightarrow s)$ otherwise.
$\bot$-rule	If 1. $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ and 2. $\bot \notin \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.

but are marked as "not present" by setting their edge labels to $\emptyset$. This has exactly the same effect as pruning.

We next prove that our calculus is sound, complete, and terminating.

Lemma 3 (Soundness). *Let $\mathcal{A}$ be an ABox and $\Xi(\mathcal{K})$ a set of DL-clauses such that $\mathcal{A} \cup \Xi(\mathcal{K})$ is satisfiable, and let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be obtained by applying a derivation rule to $\mathcal{A}$ and $\Xi(\mathcal{K})$. Then, $\mathcal{A}_i \cup \Xi(\mathcal{K})$ is satisfiable for some $1 \leq i \leq m$.*

Proof. Let I be a model of $\mathcal{A} \cup \Xi(\mathcal{K})$, and let us consider all derivation rules.

(*Hyp*-rule) Since $\sigma(U_i) \in \mathcal{A}$, we have $I \models \sigma(U_i)$ for all $1 \leq i \leq m$. But then, $I \models \sigma(V_j)$ for some $1 \leq j \leq n$. Since $\mathcal{A}_j = \mathcal{A} \cup \{\sigma(V_j)\}$, we have $I \models \mathcal{A}_j \cup \Xi(\mathcal{K})$.

($\geq$-rule) Since $\geq n\, R.C(s) \in \mathcal{A}$, we have $I \models \geq n\, R.C(s)$, which means that $\alpha_1, \ldots, \alpha_n \in \triangle^I$ exist such that $\langle s^I, \alpha_i \rangle \in R^I$ and $\alpha_i \in C^I$ for $1 \leq i \leq n$, and $\alpha_i \neq \alpha_j$ for $1 \leq i < j \leq n$. Let I' be obtained from I by setting $t_i^{I'} = \alpha_i$. Clearly, $I' \models \mathsf{ar}(R, s, t_i)$, $I' \models C(t_i)$, and $I' \models t_i \not\approx t_j$ for $i \neq j$, so $I' \models \mathcal{A}_1 \cup \Xi(\mathcal{K})$.

($\approx$-rule) Since $s \approx t \in \mathcal{A}$, we have $I \models s \approx t$, so $s^I = t^I$. Pruning removes assertions, so I is a model of the pruned ABox by monotonicity. Merging simply replaces an individual with a synonym, so, clearly, $I \models A_1 \cup \Xi(\mathcal{K})$.

($\bot$-rule) This rule is never applicable if $\mathcal{A} \cup \Xi(\mathcal{K})$ is satisfiable. □

The following corollary follows immediately from Lemma 3:

Corollary 1. *Each derivation for a satisfiable normalized $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K}$ contains a path such that $\lambda(t)$ is clash-free for each node t on the path.*

Lemma 4 (Completeness). *If a derivation for a normalized $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ contains a leaf node labeled with a clash-free ABox $\mathcal{A}'$, then $\mathcal{A} \cup \Xi(\mathcal{K})$ is satisfiable.*

Proof. We first prove the claim (*): for each ABox $\mathcal{A}'$ occurring in a derivation for $\mathcal{K}$, (i) for each $R(s,t) \in \mathcal{A}'$, t is a predecessor or a successor of s, or both s and t are named individuals, and (ii) for each $s \approx t \in \mathcal{A}'$, $s = t$, or s and t are both named, or t is a successor of a named individual and s is a named individual, or t and s have a common predecessor, or t is a successor of a successor of s. The proof is by a simple induction on the application of the derivation rules. Initially, $\mathcal{A}$ contains only named individuals. An application of the $\geq$-rule clearly preserves (*). For the $\approx$-rule, (*) holds because $\mathcal{A}'$ satisfies (ii) and merging never replaces an individual with a descendant. Finally, let us consider the *Hyp*-rule. By Definition 4, each DL-clause from $\Xi(\mathcal{K})$ is of the form (7), for A_i and B_i atomic concepts, R_i atomic roles, and C_i and D_i blocking-relevant concepts:

$$(7) \qquad \bigwedge A_i(x) \wedge \bigwedge \mathsf{ar}(R_i, x, y_i) \wedge \bigwedge B_i(y_i) \rightarrow \bigvee C_i(x) \vee \bigvee D_i(y_i) \vee \bigvee y_i \approx y_j$$

For each $y_i \approx y_j$, the antecedent contains $\mathsf{ar}(R, x, y_i) \wedge \mathsf{ar}(R, x, y_j)$. Since $\mathcal{A}'$ satisfies (i), each $\sigma(y_i)$ is either a successor or a predecessor of $\sigma(x)$ and, if $\sigma(y_i)$ is a named individual, then all $\sigma(y_j)$ are either named individuals or successors of named individuals. Thus, each $\mathcal{A}'_i$ obtained by the *Hyp*-rule satisfies (*).

We now construct a model of $\Xi(\mathcal{K}) \cup \mathcal{A}$. A *path* is a finite sequences of pairs of individuals $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. Let $\mathsf{tail}(p) = x_n$, $\mathsf{tail}'(p) = x'_n$, and $q = [p \mid \frac{x_{n+1}}{x'_{n+1}}]$ be the path $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$; we say that q is a *successor* of p, and p is a *predecessor* of q. The set of all paths $\mathcal{P}(\mathcal{A}')$ is defined inductively as follows: (i) $[\frac{a}{a}] \in \mathcal{P}(\mathcal{A}')$ for each named individual a in $\mathcal{A}'$; (ii) $[p \mid \frac{s'}{s'}] \in \mathcal{P}(\mathcal{A}')$ if $p \in \mathcal{P}(\mathcal{A}')$, s' is a successor of $\mathsf{tail}(p)$ in $\mathcal{A}'$, and s' is not blocked; and (iii) $[p \mid \frac{s}{s'}] \in \mathcal{P}(\mathcal{A}')$ if $p \in \mathcal{P}(\mathcal{A}')$, s' is a successor of $\mathsf{tail}(p)$ in $\mathcal{A}'$, and s' is directly blocked by s. For each blocking-relevant concept C and each path $p \in \mathcal{P}(\mathcal{A}')$, by the definition of blocking, $C(\mathsf{tail}(p)) \in \mathcal{A}'$ iff $C(\mathsf{tail}'(p)) \in \mathcal{A}'$; furthermore, $\mathsf{tail}(p)$ is not blocked. We denote these two properties by (**). Let I be the following interpretation:

$$\begin{aligned}
\triangle^I &= \mathcal{P}(\mathcal{A}') \\
a^I &= [\tfrac{a}{a}] \text{ for each named individual } a \text{ in } \mathcal{A}' \\
a^I &= b^I \text{ if individuals } a = c_0, c_1, \ldots, c_n = b \text{ exist such that } c_{i-1} \text{ was merged} \\
&\quad \text{into } c_i \text{ in the derivation leading to } \mathcal{A}' \\
A^I &= \{p \mid A(\mathsf{tail}(p)) \in \mathcal{A}'\} \\
R^I &= \{\langle [\tfrac{a}{a}], [\tfrac{b}{b}] \rangle \mid a \text{ and } b \textit{ are named individuals and } R(a,b) \in \mathcal{A}'\} \cup \\
&\quad \{\langle p, [p \mid \tfrac{s}{s'}] \rangle \mid s' \textit{ is a successor of } \mathsf{tail}(p) \text{ and } R(\mathsf{tail}(p), s') \in \mathcal{A}'\} \cup \\
&\quad \{\langle [p \mid \tfrac{s}{s'}], p \rangle \mid s' \textit{ is a successor of } \mathsf{tail}(p) \text{ and } R(s', \mathsf{tail}(p)) \in \mathcal{A}'\}
\end{aligned}$$

The ABox $\mathcal{A}'$ is normalized, so $\triangle^I$ is not empty. We now show that, for each $p_s = [q_s \mid \frac{s}{s'}]$ and $p_t = [q_t \mid \frac{t}{t'}]$ from $\triangle^I$, the following claims hold (***):

- *If $s' \approx t' \in \mathcal{A}'$, then $s' = t'$:* Obvious, as the $\approx$-rule is not applicable to $\mathcal{A}'$.
- *If $s' \not\approx t' \in \mathcal{A}'$, then $p_s \neq p_t$:* Since $\bot \notin \mathcal{A}'$ and the $\bot$-rule is not applicable to $s' \not\approx t'$, we have $s' \neq t'$, which implies the claim.
- *If $A(s') \in \mathcal{A}'$, then $p_s \in A^I$:* By (**), we have $A(s) \in \mathcal{A}'$, so $p_s \in A^I$.
- *If $\neg A(s') \in \mathcal{A}'$, then $p_s \notin A^I$.* Since $\bot \notin \mathcal{A}'$ and the $\bot$-rule is not applicable to $\neg A(s')$, we have $A(s') \notin \mathcal{A}'$. By (**), this implies $A(s) \notin \mathcal{A}'$, so $p_s \notin A^I$.

– *If* $\geq n\, R.C(s') \in \mathcal{A}'$, *then* $p_s \in (\geq n\, R.C)^I$*:* By (**), $\geq n\, R.C(s) \in \mathcal{A}'$ and s is not blocked. The $\geq$-rule is not applicable to $\geq n\, R.C(s)$, so individuals $u_1, \ldots, u_n$ exist such that $\mathsf{ar}(R, s, u_i) \in \mathcal{A}'$ and $C(u_i) \in \mathcal{A}'$ for $1 \leq i \leq n$, and $u_i \not\approx u_j \in \mathcal{A}'$ for $1 \leq i < j \leq n$. By (*), these possibilities exist for each u_i:

  - u_i is a successor of s. If u_i is directly blocked by u_i', let $p_{u_i} = [p_s \mid \frac{u_i'}{u_i}]$; otherwise, let $p_{u_i} = [p_s \mid \frac{u_i}{u_i}]$.
  - u_i is a predecessor of s. Let $p_{u_i} = q_s$. If $\mathsf{tail}'(p_{u_i}) \neq u_i$, this is because s' is blocked, but then, by the conditions of blocking, $C(\mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$ and $\mathsf{ar}(R, s', \mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$.
  - u_i is neither a predecessor nor a successor of s. Then, both s and u_i are named individuals, so let $p_{u_i} = [\frac{u_i}{u_i}]$.

  In all cases, we have $\mathsf{ar}(R, s', \mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$, which implies $\langle p_s, p_{u_i} \rangle \in R^I$, and $C(\mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$, which implies $p_{u_i} \in C^I$. Consider now each pair of paths p_{u_i} and p_{u_j} with $i \neq j$. If $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \in \mathcal{A}'$, then clearly $p_{u_i} \neq p_{u_j}$. If $\mathsf{tail}'(p_{u_i}) \not\approx \mathsf{tail}'(p_{u_j}) \notin \mathcal{A}'$, this is because $\mathsf{tail}'(p_{u_i}) \neq u_i$, which is possible only if s' is directly blocked by s and u_i is a predecessor of s. Since s can have at most one predecessor, no u_j with $j \neq i$ is a predecessor of s, so $p_{u_i} \neq p_{u_j}$. Thus, we conclude that $p_s \in (\geq n\, R.C)^I$.

Clearly, (***) implies that $I \models \alpha'$ for each assertion $\alpha' \in \mathcal{A}'$ that contains only named individuals. Consider now each $\alpha \in \mathcal{A}$. If $\alpha \notin \mathcal{A}'$, then some named individuals in α were merged into other individuals; but then, $\mathcal{A}'$ contains the assertion α' obtained by this merging, so $I \models \alpha$ by the definition of I.

It remains to be shown that $I \models \Xi(\mathcal{K})$. Consider each DL-clause $r \in \Xi(\mathcal{K})$ of the form (7) and each variable mapping μ. Let $p_x = \mu(x)$, $p_{y_i} = \mu(y_i)$, and $s' = \mathsf{tail}'(p_x)$. Assume now that each atom from the antecedent of r is true in I and μ—that is, $p_x \in A_i^I$, $p_{y_i} \in B_i^I$, and $\langle p_x, p_{y_i} \rangle \in R^I$.

If s' is not blocked, let $s = s'$ and $t_i = \mathsf{tail}'(p_{y_i})$. By the definition of I, we have $A_i(s) \in \mathcal{A}'$, $B_i(t_i) \in \mathcal{A}'$, and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$.

If s' is blocked, let $s = \mathsf{tail}(p_x)$; that is, s is the individual that blocks s'. By the definition of I, since $p_x \in A_i^I$, we have $A_i(s) \in \mathcal{A}'$. If $\mathsf{tail}'(p_{y_i})$ is a successor of s, let $t_i = \mathsf{tail}'(p_{y_i})$; now $p_{y_i} \in B_i^I$ and $\langle p_x, p_{y_i} \rangle \in R_i^I$ imply $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$. If $\mathsf{tail}'(p_{y_i})$ is not a successor of s, let t_i be the predecessor of s; this predecessor exists by the definition of blocking. Furthermore, $p_{y_i} \in B_i^I$ and $\langle p_x, p_{y_i} \rangle \in R_i^I$ imply $B_i(\mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s', \mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$; by the definition of blocking, we have $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i, s, t_i) \in \mathcal{A}'$ as well.

Let σ be a mapping such that $\sigma(x) = s$ and $\sigma(y_i) = t_i$. The *Hyp*-rule is not applicable to $\mathcal{A}'$, so some of the atoms from the consequent of $\sigma(r)$ must be present in $\mathcal{A}'$. Assume first that $C_i(s) \in \mathcal{A}'$ or $D_i(t_i) \in \mathcal{A}'$. By the definition of blocking, $C_i(\mathsf{tail}'(p_x)) \in \mathcal{A}'$ or $D_i(\mathsf{tail}'(p_{y_i})) \in \mathcal{A}'$, respectively; by (***), this implies $p_x \in C_i^I$ or $p_{y_i} \in D_i^I$, respectively. Assume now that $t_i \approx t_j \in \mathcal{A}'$. By (***), we have $t_i = t_j$. If p_{y_i} and p_{y_j} are both successors of p_x, then $t_i = \mathsf{tail}'(p_{y_i})$ and $t_j = \mathsf{tail}'(p_{y_j})$, so $t_i = t_j$ implies $p_{y_i} = p_{y_j}$. If p_{y_i} and p_{y_j} are both predecessors of p_x, we have $p_{y_i} = p_{y_j}$ since p_x can have at most one predecessor. Finally, let us assume that p_{y_i} is a predecessor of p_x, which is a predecessor of p_{y_j}. Then,

$\mathsf{tail}'(p_{y_j}) = t_j$; furthermore, since t_i is not blocked, we have $\mathsf{tail}'(p_{y_j}) \neq t_i$, which contradicts the assumption that $t_i = t_j$. □

If $\prec$ coincides with the descendant relationship, the termination proof is analogous to [10, Lemma 3], so we present here only the intuition. Consider any ABox $\mathcal{A}$ in the derivation. There are at most exponentially many different tuples $\langle \mathcal{L}_\mathcal{A}(s), \mathcal{L}_\mathcal{A}(s,t), \mathcal{L}_\mathcal{A}(t,s), \mathcal{L}_\mathcal{A}(t)\rangle$, so an individual can have at most exponentially many nonblocked ancestors. Thus, $\mathcal{A}$ can be viewed as a tree with exponential depth and a linear branching factor, so the number of nonblocked individuals is at most doubly exponential. When an individual s becomes blocked, at most double exponentially many nonblocked descendants of s can become indirectly blocked, so $|\mathcal{A}|$ is at most doubly exponential in $|\mathcal{K}|$. Due to pruning, the $\geq$-rule can be applied to an individual at most $|\mathcal{K}|$ times. We construct the derivation nondeterministically, so our algorithm runs in 2NExpTime.

If $\prec$ is total, the number of nonblocked individuals in $\mathcal{A}$ is exponential. Analogously to the previous case, we can conclude that the number of individuals is at most exponential, so our algorithm runs in NExpTime. The DL $\mathcal{SHIQ}$ is ExpTime-complete [2], so our algorithm is not worst-case optimal. Worst-case optimal tableau algorithms for fragments of $\mathcal{SHIQ}$ have been presented in [7, 4]. These algorithms use caching, which is related to anywhere blocking; however, to obtain the desired complexity result, they use cuts and are thus unlikely to be practical. Furthermore, we are not aware of any practical implementation of these calculi. On Horn-$\mathcal{SHIQ}$ [12] knowledge bases, however, our algorithm is deterministic, so it runs in ExpTime. It is known that Horn-$\mathcal{SHIQ}$ is ExpTime-hard [13], so our algorithm gives a worst-case optimal decision procedure.

Lemma 5 (Termination, [10]). *For a normalized $\mathcal{ALCHIQ}$ knowledge base $\mathcal{K}$, every derivation from $\mathcal{K}$ is finite.*

Lemmas 1, 4, 5, and Corollary 1 immediately imply our main theorem:

Theorem 1. *A $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ is satisfiable if and only if each derivation from $\mathcal{K}' = \Delta(\Omega(\mathcal{K}))$ contains a leaf node t such that $\lambda(t)$ is clash-free; furthermore, the construction of each such derivation terminates.*

4.3 Applying the Algorithm to Other DLs

For DLs with inverse roles but without number restrictions, traditional tableau algorithms can use simpler *equality blocking* [9]: an unnamed individual s is blocked by an individual s' in $\mathcal{A}$ iff $s' \prec s$ and $\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(s')$. Such blocking must be applied with care in our setting. Consider the knowledge base (8), on which our algorithm produces the ABox (10).

$$\mathcal{K}_4 = \{C \sqsubseteq \exists R.D,\ \ D \sqsubseteq \exists S^-.C,\ \ \top \sqsubseteq \forall R.\bot \sqcup \forall S.\bot,\ \ C(a)\} \tag{8}$$

$$\Xi(\mathcal{K}_4) = \{C(x) \rightarrow \exists R.D(x),\ D(x) \rightarrow \exists S^-.C(x),\ R(x,y_1) \wedge S(x,y_2) \rightarrow \bot\} \tag{9}$$

$$\mathcal{A}_4 = \left\{ \begin{matrix} C(a), \\ \exists R.D(a), \end{matrix} R(a,a.1),\ \begin{matrix} D(a.1), \\ \exists S^-.C(a.1), \end{matrix} S(a.1.1, a.1),\ \begin{matrix} C(a.1.1), \\ \exists R.D(a.1.1) \end{matrix} \right\} \tag{10}$$

The individual $a.1.1$ is directly blocked by a, so the algorithm terminates; an expansion of $\exists R.D(a.1.1)$, however, would reveal that $\mathcal{K}$ is unsatisfiable. The problem arises because the DL-clause $R(x, y_1) \wedge S(x, y_2) \rightarrow \bot$ contains two role atoms, which allows it to examine both the successors and the predecessor of x. Equality blocking, however, does not ensure that both predecessors and successors of x have been fully built. We can correct this problem by requiring the normalized GCIs to contain at most one $\forall R.C$ concept. For example, if we replace our DL-clause with $R(x, y_1) \rightarrow Q(x)$ and $Q(x) \wedge S(x, y_2) \rightarrow \bot$, then the first DL-clause additionally derives $Q(a)$, so $a.1.1$ is not blocked.

For DLs without inverse roles, tableau algorithms typically use *subset blocking* [1]: an unnamed individual s is blocked by s' in $\mathcal{A}$ iff $s' \prec s$ and $\mathcal{L}_{\mathcal{A}}(s) \subseteq \mathcal{L}_{\mathcal{A}}(s')$. Subset blocking is not applicable in our setting. Consider the knowledge base (11), on which our algorithm produces the ABox (13):

$$\mathcal{K}_5 = \{C \sqsubseteq \exists R.C,\ \ C \sqsubseteq \exists S.D,\ \ \exists S.D \sqsubseteq E,\ \ \exists R.E \sqsubseteq \bot,\ \ C(a)\} \tag{11}$$

$$\Xi(\mathcal{K}_5) = \left\{ \begin{array}{ll} C(x) \rightarrow \exists R.C(x), & C(x) \rightarrow \exists S.D(x), \\ S(x, y) \wedge D(y) \rightarrow E(x), & R(x, y) \wedge E(y) \rightarrow \bot \end{array} \right\} \tag{12}$$

$$\mathcal{A}_5 = \left\{ \begin{array}{llllll} C(a), & \exists S.D(a), & S(a, a.1), & D(a.1), & & E(a), \\ \exists R.C(a), & R(a, a.2), & C(a.2), & \exists R.C(a.2), & \exists S.D(a.2) & \end{array} \right\} \tag{13}$$

Now $a.2$ is directly blocked by a. If, however, we expanded $\exists S.D(a.2)$ into $S(a.2, a.2.1)$ and $D(a.2.1)$, we can derive $E(a.2)$; together with $R(a, a.2)$ and the last DL-clause from $\Xi(\mathcal{K}_5)$, we get a contradiction. Even without inverse roles, DL-clauses can propagate information from successors to predecessors.

5 Implementation

Based on the calculus from Section 4, we have implemented a prototype DL reasoner.[1] Currently, it can only handle Horn DL-clauses—our main goal was to show that significant performance improvements can be gained by exploiting the deterministic nature of many ontologies.

To classify a knowledge base $\mathcal{K}$, we run our algorithm on $\mathcal{K}_i = \mathcal{K} \cup \{C_i(a_i)\}$ for each concept C_i, obtaining an ABox $\mathcal{A}_i$. If $D(a_i) \in \mathcal{A}_i$ and $D(a_i)$ was derived without making any nondeterministic choices, then $\mathcal{K} \models C_i \sqsubseteq D$. Since our test ontologies are translated to Horn DL-clauses on which our algorithm is deterministic, $D(a_i) \in \mathcal{A}_i$ iff $\mathcal{K} \models C_i \sqsubseteq D$. Thus, we can classify $\mathcal{K}$ with a linear number of calls to our algorithm. This optimization is also applicable in standard tableau calculi; the nondeterministic handling of GCIs, however, diminishes its value.

We also employ the following optimization: when applying the calculus to $\mathcal{K}_i$, we use the nonblocked individuals from Γ_i as potential blockers, where Γ_i is the union of all satisfiable ABoxes $\mathcal{A}_j$ for $j < i$. Namely, assume that we run our algorithm on $\mathcal{K}'_i = \mathcal{K} \cup \{C_i(a_i)\} \cup \Gamma_i$, where a_i is fresh. In $\mathcal{SHIQ}$, a_i cannot interact with Γ_i; furthermore Γ_i is satisfiable, so $\mathcal{K} \models C_i \sqsubseteq D$ iff our

[1] http://www.cs.man.ac.uk/∼bmotik/HermiT/

Table 5. Results of Performance Evaluation

Ontology	HT	HT-anc	Pellet	FaCT++	Racer
NCI	8 s	9 s	44 min	32 s	36 s
GALEN original	44 s	—	—	—	—
GALEN simplified	7 s	104 s	—	859 s	—

algorithm derives $D(a_i)$ from $\mathcal{K}'_i$. Thus, due to anywhere blocking, we can use the nonblocked individuals from Γ_i as blockers without affecting the correctness of our algorithm. This optimization is applicable even if nondeterministic choices were made in deriving $\mathcal{A}_j$, it is easy to implement, and, like *caching* [2, Chapter 9], it greatly reduces the time needed to classify an ontology since it prevents the computation of the same subtrees in different runs.

Table 5 shows the times that our reasoner, Pellet 1.3, FaCT++ 1.1.4, and Racer 1.9.0 take to classify our test ontologies. To isolate the improvements due to each of the two innovations of our algorithm, we evaluated our system with anywhere blocking (denoted as HT), as well as with ancestor blocking [10] (denoted as HT-anc). All ontologies are available from our reasoner's Web page.

NCI is a relatively large (about 23000 atomic concepts) but simple ontology. FaCT++ and RACER can classify NCI in a short time mainly due to an optimization which eliminates many unnecessary tests, and the fact that all axioms in NCI are definitional so they are handled efficiently by absorption. We conjecture that Pellet is slower by two orders of magnitude because it does not use these optimizations, so it must deal with disjunctions.

GALEN has often been used as a benchmark for DL reasoning. The original version of GALEN contains about 2700 atomic concepts and many GCIs similar to (2). Most GCIs cannot be absorbed without any residual nondeterminism. Thus, the ontology is hard because it requires the generation of large models with many nondeterministic choices. Hence, GALEN has been simplified by removing 273 axioms, and this simplified version of GALEN has commonly been used for performance testing. As Table 5 shows, only HT can classify the original version of GALEN. In particular, anywhere blocking prevents our reasoner from generating the same fragments of a model in different branches.

6 Conclusion

In this paper, we presented a novel reasoning algorithm for DLs that combines hyper-inferences to reduce the nondeterminism due to GCIs with anywhere blocking to reduce the sizes of generated models. In future, we shall extend our reasoner to handle disjunction and conduct a more comprehensive performance evaluation. Furthermore, we shall investigate the possibilities of optimizing the blocking condition and heuristically guiding the model construction to further reduce the sizes of the models created. Finally, we shall try to extend our approach to the DLs $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, which provide the logical underpinning of the Semantic Web ontology languages.

References

1. F. Baader, M. Buchheit, and B. Hollunder. Cardinality Restrictions on Concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
3. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. JELIA '96*, pages 1–17, Évora, Portugal, September 30–October 3 1996.
4. F. M. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.
5. C. Fermüller, T. Tammet, N. Zamov, and A. Leitsch. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI*. Springer, 1993.
6. C. G. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution Decision Procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1791–1849. Elsevier Science, 2001.
7. R. P. Goré and L. Nguyen. EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies. In *Proc. TABLEAUX 2007*, Aix en Provence, France, July 3–6 2007. Springer. To appear.
8. V. Haarslev and R. Möller. RACER System Description. In *Proc. IJCAR 2001*, pages 701–706, Siena, Italy, June 18–23 2001.
9. I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
10. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In *Proc. CADE-17*, pages 482–496, Pittsburgh, USA, 2000.
11. A. K. Hudek and G. Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In *Proc. DL 2006*, Windermere, UK, May 30-June 1 2006.
12. U. Hustadt, B. Motik, and U. Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proc. IJCAI 2005*, pages 466–471, Edinburgh, UK, July 30–August 5 2005.
13. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity Boundaries for Horn Description Logics. In *Proc. AAAI 2007*, Vancouver, BC, Canada, July 22-26 2007. AAAI Press. To appear.
14. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.
15. B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner. Poster, In Proc. ISWC 2004, Hiroshima, Japan, November 7–11, 2004.
16. D. A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Logic and Computation*, 2(3):293–304, 1986.
17. A. Robinson. Automatic Deduction with Hyper-Resolution. *Int. Journal of Computer Mathematics*, 1:227–234, 1965.
18. R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In *Proc. CADE-19*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003.
19. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.
20. D. Tsarkov and I. Horrocks. Efficient Reasoning with Range and Domain Constraints. In *Proc. DL 2004*, Whistler, BC, Canada, June 6–8 2004.
21. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. IJCAR 2006*, pages 292–297, Seattle, WA, USA, 2006.