Hypertableau Reasoning for Description Logics

Boris Motik [*], Rob Shearer, Ian Horrocks

University of Oxford, UK

Abstract

We present a novel reasoning calculus for the Description Logic $\mathcal{SHOIQ}^+$—a knowledge representation formalism 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*. We also present a new way of ensuring termination while allowing for nomials, inverse roles, and number restrictions—a combination of DL constructs that has proven notoriously difficult to handle. 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.

Key words: description logics, hypertableau, OWL, Semantic Web

1 Introduction

Description Logics (DLs) [3] are a family of knowledge representation formalisms with well-understood formal properties. DLs have been applied to numerous problems in computer science. Recent interest in DLs has been spurred by their application in the Semantic Web: the DL $\mathcal{SHOIQ}$ provides the logical underpinning for the Web Ontology Language (OWL), and the DL $\mathcal{SROIQ}$ [27] is used in OWL 2—an extension of OWL currently being standardized by the W3C.

A central component of most DL applications is an efficient and scalable reasoner. Modern reasoners, such as Pellet [31], FaCT++ [41], and RACER [18], are typically

[⋆] This is an extended version of a paper published at CADE 2007 [30].

[*] Corresponding author.

Email addresses: `boris.motik@comlab.ox.ac.uk` (Boris Motik), `rob.shearer@comlab.ox.ac.uk` (Rob Shearer), `ian.horrocks@comlab.ox.ac.uk` (Ian Horrocks).

based on tableau calculi [3, Chapter 2], which 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 [3, Chapter 9]. Despite major advances in tableau reasoning algorithms, ontologies are still encountered in practice that cannot be handled by existing DL reasoners. Two main sources of inefficiency have been identified in literature [3, Chapter 3].

Or-branching is one source of inefficiency in tableau algorithms: given a disjunctive assertion $(C \sqcup D)(s)$, a tableau algorithm nondeterministically guesses that either $C(s)$ or $D(s)$ is true. 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 $D(s)$, which can give rise to exponential behavior. General concept inclusions (GCIs)—implications 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 [3, Chapter 9][24,39] reduce the 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.

And-branching is another source of inefficiency in tableau algorithms: the expansion of a model due to existential quantifiers can generate very large models. 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 $\mathcal{SHOIQ}^+$, which is obtained by extending $\mathcal{SHOIQ}$ with local reflexivity and disjoint, reflexive, irreflexive, symmetric, and asymmetric roles. The main difference between $\mathcal{SHOIQ}^+$ and $\mathcal{SROIQ}$ is that the latter provides for *generalized role inclusions* of the form $R_1 \circ \ldots \circ R_n \sqsubseteq R$. We conjecture that generalized role inclusions can be encoded using standard GCIs as proposed by Demri and de Nivelle [10]; thus, by adding a suitable preprocessing phase, the results from this paper should allow us to handle $\mathcal{SROIQ}$ and OWL 2 as well.

Our algorithm first preprocesses a $\mathcal{SHOIQ}^+$ knowledge base 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 DL-clause body can be matched to already derived consequences. 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 [6] and hyperresolution [34] calculi.

Hyperresolution decides many fragments of first-order logic (see, e.g., [13,12] for an overview), as well as various description and modal logics (e.g., [15,26]). Unlike most of these fragments, $\mathcal{SHOIQ}^+$ 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 [23] to detect cyclic computations. Due to hyper-inferences, the soundness and correctness proofs from [23] do not carry over immediately 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 [23] to *anywhere pairwise blocking*: an individual can be blocked by an individual that is not necessarily an ancestor, which can reduce the sizes of the constructed models. Anywhere blocking has already been used with *subset* blocking [2]; however, to the best of our knowledge, it has been neither used with the more sophisticated pairwise blocking nor tested in practice.

Ensuring termination of a tableau decision procedure for $\mathcal{SHOIQ}$ is known to be notoriously difficult, mainly due to a certain interaction between nominals, inverse roles, and number restrictions. This problem has been finally solved in [21] by extending the tableau calculus with a *nominal generation* rule. In certain situations, this rule guesses and introduces a new nominals, and is thus a potential source of inefficiency in practice. In this paper, we present a simpler and more efficient rule that, we believe, can be implemented more easily.

We have implemented our calculus in a new reasoner called HermiT. [1] 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 description logic $\mathcal{SHOIQ}^+$ is obtained from $\mathcal{SROIQ}$ [27] by disallowing general role inclusions, and it is defined as follows. A *signature* is a triple $\Sigma = (N_R, N_C, N_I)$ consisting of mutually disjoint sets of *atomic roles* N_R, *atomic concepts* N_C, and *individuals* N_I. The set of *roles* is then $N_R \cup \{R^- \mid R \in N_R\}$. For $R \in N_R$, let $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$. An *RBox* $\mathcal{R}$ is a finite set of axioms of the form $R_1 \sqsubseteq R_2$ (*role inclusions*), $\mathsf{Dis}(S_1, S_2)$ (*role disjointness*), $\mathsf{Ref}(R)$ (*reflexivity*), $\mathsf{Irr}(S)$ (*irreflexivity*), $\mathsf{Sym}(R)$ (*symmetry*), $\mathsf{Asy}(S)$ (*asymmetry*), and $\mathsf{Tra}(R)$ (*transitivity*), where R, R_1, and R_2 are roles, and S, S_1, and S_2 are *simple* roles, as defined next. Let $\sqsubseteq^*_{\mathcal{R}}$ be the reflexive-transitive closure of the relation $\{R_1 \sqsubseteq R_2, \mathsf{Inv}(R_1) \sqsubseteq \mathsf{Inv}(R_2) \mid R_1 \sqsubseteq R_2 \in \mathcal{R}\}$. A role R is *transitive* in $\mathcal{R}$ if a role R' exists such that $R' \sqsubseteq^*_{\mathcal{R}} R$, $R \sqsubseteq^*_{\mathcal{R}} R'$, and either $\mathsf{Tra}(R') \in \mathcal{R}$ or $\mathsf{Tra}(\mathsf{Inv}(R')) \in \mathcal{R}$. A role S is *simple* if no transitive role R exists such that $R \sqsubseteq^*_{\mathcal{R}} S$. The set of *concepts* is the smallest set containing $\top$ (the *top concept*), $\bot$ (the *bottom concept*), A (*atomic concept*), $\{a\}$ (*nominal*), $\neg C$ (*negation*), $C \sqcap D$ (*conjunction*), $C \sqcup D$

[1] `http://web.comlab.ox.ac.uk/people/boris.motik/HermiT/`

Table 1
Model-Theoretic Semantics of $\mathcal{SHOIQ}^+$

Semantics of Roles and Concepts	Semantics of Axioms
$(R^-)^I = \{\langle y,x\rangle \mid \langle x,y\rangle \in R^I\}$	$C \sqsubseteq D \quad \Rightarrow C^I \subseteq D^I$
$\top^I = \triangle^I$	$R_1 \sqsubseteq R_2 \quad \Rightarrow R_1^I \subseteq R_2^I$
$\bot^I = \emptyset$	$\mathsf{Dis}(S_1,S_2) \Rightarrow S_1^I \cap S_2^I = \emptyset$
$\{s\}^I = \{s^I\}$	$\mathsf{Ref}(R) \quad \Rightarrow \forall x \in \triangle^I : \langle x,x\rangle \in R^I$
$(\neg C)^I = \triangle^I \setminus C^I$	$\mathsf{Irr}(S) \quad \Rightarrow \forall x \in \triangle^I : \langle x,x\rangle \notin S^I$
$(C \sqcap D)^I = C^I \cap D^I$	$\mathsf{Sym}(R) \quad \Rightarrow R^I \subseteq (\mathsf{Inv}(R))^I$
$(C \sqcup D)^I = C^I \cup D^I$	$\mathsf{Asy}(S) \quad \Rightarrow S^I \cap (\mathsf{Inv}(S))^I = \emptyset$
$(\exists R.C)^I = \{x \mid \exists y : \langle x,y\rangle \in R^I \wedge y \in C^I\}$	$\mathsf{Tra}(R) \quad \Rightarrow (R^I)^+ \subseteq R^I$
$(\forall R.C)^I = \{x \mid \forall y : \langle x,y\rangle \in R^I \rightarrow y \in C^I\}$	$C(a) \quad \Rightarrow a^I \in C^I$
$(\exists S.\mathsf{Self})^I = \{x \mid \langle x,x\rangle \in S^I\}$	$R(a,b) \quad \Rightarrow \langle a^I, b^I\rangle \in R^I$
$(\geq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x,y\rangle \in S^I \wedge y \in C^I\} \geq n\}$	$a \approx b \quad \Rightarrow a^I = b^I$
$(\leq n\, S.C)^I = \{x \mid \sharp\{y \mid \langle x,y\rangle \in S^I \wedge y \in C^I\} \leq n\}$	$a \not\approx b \quad \Rightarrow a^I \neq b^I$

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

(*disjunction*), $\exists R.C$ (*existential restriction*), $\forall R.C$ (*universal restriction*), $\exists S.\mathsf{Self}$ (*local reflexivity*), $\geq n\, S.C$ (*at-least restriction*), and $\leq n\, S.C$ (*at-most restriction*), for A an atomic concept, a an individual, 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 C and D concepts. An *ABox* $\mathcal{A}$ is a finite set of assertions of the form $C(a)$ (*concept assertion*), $R(a,b)$ (*role assertion*), $a \approx b$ (*equality assertion*), and $a \not\approx b$ (*inequality assertion*), where C is a concept, R is a role, and a and b are individuals. A $\mathcal{SHOIQ}^+$ 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 $\mathcal{SHOIQ}^+$ is checking *satisfiability* of $\mathcal{K}$—that is, checking whether a model of $\mathcal{K}$ exists. A concept C *subsumes* a concept D, written $\mathcal{K} \models C \sqsubseteq D$, if $C^I \subseteq D^I$ for each model I of $\mathcal{K}$. It is well known that $\mathcal{K} \models C \sqsubseteq D$ if and only if $\mathcal{K} \cup \{(C \sqcap \neg D)(a)\}$ is unsatisfiable, where a is an individual that does not occur in $\mathcal{K}$ [3].

The *negation-normal form* $\mathsf{nnf}(C)$ of a concept C is the concept equivalent to C in which negations occur only in front of atomic concepts and concepts of the form $\{a\}$ and $\exists S.\mathsf{Self}$. The concept $\mathsf{nnf}(C)$ can be computed in time linear in the size of C [3]. Let $\dot{\neg} C = \mathsf{nnf}(\neg C)$. With $|\mathcal{K}|$ we denote the size of $\mathcal{K}$—that is, the number of symbols required to encode $\mathcal{K}$ on the input tape of a Turing machine (numbers can be coded in binary). The DL $\mathcal{ALCHOIQ}^+$ is obtained from $\mathcal{SHOIQ}^+$ by disallowing transitivity axioms. $\mathcal{SHOIQ}$ is obtained from $\mathcal{SHOIQ}^+$ by disallowing local reflexivity, role disjointness, reflexivity, irreflexivity, symmetry, and asymmetry. $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHIO}$ are obtained from $\mathcal{SHOIQ}$ by disallowing transitive roles, nominals, inverse roles, and number restrictions, respectively.

3 Motivation and Algorithm Overview

In this section, we present an overview of the main aspects of our algorithm. We explain in Section 3.1 the roots of scalability problems in standard tableau algorithms, and in Section 3.2 we outline the way in which we address them.

3.1 Causes of Scalability Problems in Tableau Algorithms

To show that a knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is satisfiable, a tableau algorithm constructs a sequence of ABoxes $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$ called a *derivation*, where each $\mathcal{A}_i$ is obtained from $\mathcal{A}_{i-1}$ by an application of one *inference rule.*[2] The inference rules make the information implicit in the axioms of $\mathcal{R}$ and $\mathcal{T}$ explicit, and thus evolve the ABox $\mathcal{A}$ towards a (representation of a) model of $\mathcal{K}$. The algorithm terminates either if no inference rule is applicable to some $\mathcal{A}_n$, in which case $\mathcal{A}_n$ represents a model of $\mathcal{K}$, or if $\mathcal{A}_n$ contains an obvious contradiction, in which case the model construction has failed. The following inference rules are commonly used in DL tableau calucli.

- $\sqcup$-rule: Given $(C_1 \sqcup C_2)(s)$, derive either $C_1(s)$ or $C_2(s)$.
- $\sqcap$-rule: Given $(C_1 \sqcap C_2)(s)$, derive $C_1(s)$ and $C_2(s)$.
- $\exists$-rule: Given $(\exists R.C)(s)$, derive $R(s,t)$ and $C(t)$ for t a fresh individual.
- $\forall$-rule: Given $(\forall R.C)(s)$ and $R(s,t)$, derive $C(t)$.
- $\sqsubseteq$-rule: Given a GCI $C \sqsubseteq D$ and an individual s, derive $(\neg C \sqcup D)(s)$.

The $\sqcup$-rule is nondeterministic: if $(C_1 \sqcup C_2)(s)$ is true, then $C_1(s)$ or $C_2(s)$ or both are true. Therefore, tableau calculi make a nondeterministic guess and choose either C_1 or C_2; if choosing C_1 leads to a contradiction, the algorithm must backtrack and try C_2. Thus, $\mathcal{K}$ is unsatisfiable only if all choices fail to construct a model. We next discuss two sources of complexity inherent in the tableau inference rules.

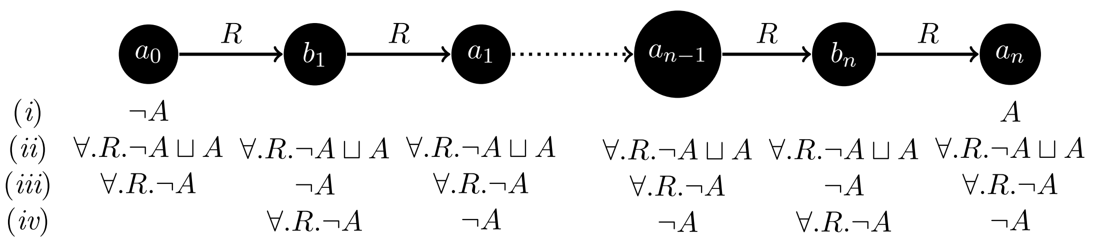
3.1.1 Or-Branching

Handing disjunctions through reasoning by case is often called *or-branching*, and it is a major source of inefficiency [3]. The $\sqsubseteq$-rule is the main source of or-branching, as it adds a disjunction for each GCI to each individual in an ABox. Consider the following knowledge base consisting of the TBox $\mathcal{T}_1$ and the ABox $\mathcal{A}_1$.

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

The ABox $\mathcal{A}_1$ is graphically shown in Figure 1. The individuals occurring in the ABox are represented as black dots, an assertion of the form $A(a_0)$ is represented by placing A next to the individual a_0, and an assertion of the form $R(a_0, b_1)$ is

[2] Some formalizations of tableau algorithms work on *completion graphs*, which have a natural correspondence to ABoxes.



a_0 $\xrightarrow{R}$ b_1 $\xrightarrow{R}$ a_1 ⋯ a_{n-1} $\xrightarrow{R}$ b_n $\xrightarrow{R}$ a_n
(i) $\neg A$ A
(ii) $\forall.R.\neg A \sqcup A$ $\forall.R.\neg A \sqcup A$ $\forall.R.\neg A \sqcup A$ $\forall.R.\neg A \sqcup A$ $\forall.R.\neg A \sqcup A$ $\forall.R.\neg A \sqcup A$
(iii) $\forall.R.\neg A$ $\neg A$ $\forall.R.\neg A$ $\forall.R.\neg A$ $\neg A$ $\forall.R.\neg A$
(iv) $\forall.R.\neg A$ $\neg A$ $\neg A$ $\forall.R.\neg A$ $\neg A$

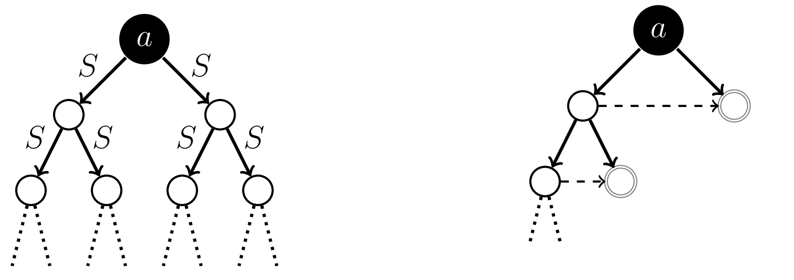
Fig. 1. Or-Branching Example

represented as an R-labeled arrow from a_0 to b_1. Initially, $\mathcal{A}_1$ contains only the concept assertions shown in line (i).

To satisfy the GCI in $\mathcal{T}_1$, a tableau algorithm applies the $\sqsubseteq$-rule, thus adding the assertions shown in line (ii) of Figure 1. Tableau algorithms usually have a significant degree of freedom in choosing the order in which to process the assertions in an ABox; in fact, finding an order that exhibits good performance in practice requires advanced heuristics [40]. Let us assume that the algorithm chooses to process the assertions on a_i before those on b_j. Hence, by applying the rules to all a_i, the algorithm derives the assertions shown in line (iii) of Figure 1; after that, by applying the rules to all b_i, the algorithm derives the assertions shown in line (iv) of Figure 1. The ABox now contains both $A(a_n)$ and $\neg A(a_n)$, which is a contradiction. Thus, the algorithm needs to backtrack its most recent choice, so it 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, changes the guess on a_n to $A(a_n)$, and repeats the work for all b_i. 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. None of the standard backtracking optimizations [3, Chapter 9] are helpful: 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 for $i \neq j$.

The GCI $\exists R.A \sqsubseteq A$, however, is not inherently nondeterministic: it is equivalent to the Horn clause $\forall x, y : [R(x, y) \wedge A(y) \rightarrow A(x)]$, which can be applied in a bottom-up manner to derive the assertions $A(b_n), A(a_{n-1}), \ldots, A(a_0)$ and reveal 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 processing of GCIs in tableau algorithms can be "unnecessarily" nondeterministic.

Various *absorption* optimizations [3, Chapter 9] have been developed to address this problem. The basic absorption algorithm tries to rewrite GCIs into the form $B \sqsubseteq C$ where B is an atomic concept. Then, instead of deriving $\neg B \sqcup C$ for each individual in an ABox, $C(s)$ is derived only if the ABox contains $B(s)$; thus, the absorbed GCIs can be applied in a "more deterministic" way. This technique has been extended in several ways. *Role absorption* [39] rewrites GCIs into the form



(a) Ancestor Blocking (b) Anywhere Blocking

Fig. 2. And-Branching Example

$\exists R.\top \sqsubseteq C$; then, $C(s)$ is derived only if an ABox contains $R(s,t)$. *Binary absorption* [24] rewrites GCIs into the form $B_1 \sqcap B_2 \sqsubseteq C$; then, $C(s)$ is derived only if an ABox contains both $B_1(s)$ and $B_2(s)$. Neither of these two optimizations, however, help us deal with the GCI in (1) directly. Role absorption produces an axiom $\exists R.\top \sqsubseteq A \sqcap \forall R.\neg A$, which still contains a disjunction in the consequent. Furthermore, binary absorption is not applicable to (1), since the axiom does not contain two concepts on the left-hand side of the implication symbol $\sqsubseteq$. The axiom (1) can be absorbed if it is rewritten as $A \sqsubseteq \forall R^-.A$. In practice, however, it is often unclear in advance which combination of transformation and absorption techniques will yield the best results. Therefore, absorption algorithms are in practice guided primarily by heuristics and may not eliminate all nondeterminism.

3.1.2 And-Branching

The introduction of new individuals in the $\exists$-rule is often called *and-branching*, and it is another major source of inefficiency in tableau algorithms [3]. Consider the following (satisfiable) knowledge base $\mathcal{K}_2$.

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

At-least restrictions are dealt with in tableau algorithms by the $\geq$-rule, which is quite similar to the $\exists$-rule: from $\geq n\, R.C(s)$, the $\geq$-rule derives $R(s,t_i)$ and $C(t_i)$ for $1 \leq i \leq n$, and $t_i \not\approx t_j$ for $1 \leq i < j \leq n$. Thus, the fact $A_1(a)$ implies existence of at least two individuals in A_2, which imply existence of at least two individuals in A_3, and so on. A tableau algorithm thus constructs on $\mathcal{K}_2$ a binary tree, shown in Figure 2a, in which with each individual is labeled with some A_i and an element of $\Pi = \{B_1, C_1\} \times \ldots \times \{B_m, C_m\}$. Each individual at depth n is an instance of A_n; because of the GCI $A_n \sqsubseteq A_1$, this individual must be an instance of A_1 as well, so we can repeat the whole construction and generate an even deeper tree. Clearly, a naïve application of the tableau rules does not terminate if the TBox contains existential quantifiers in cycles.

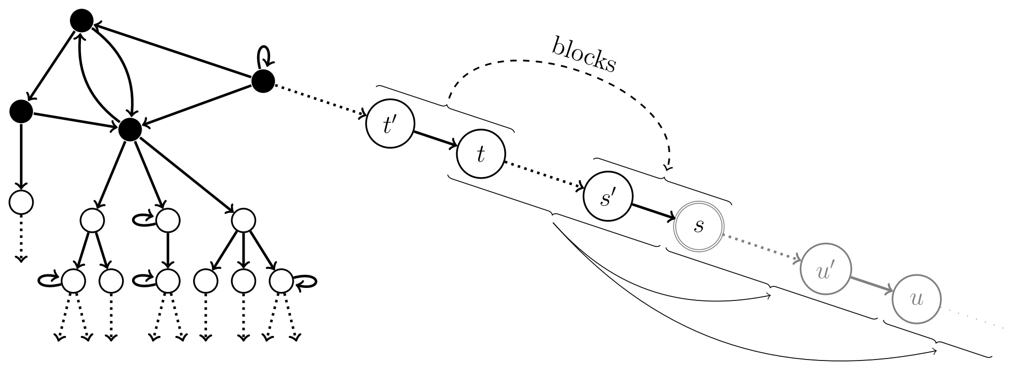


Fig. 3. Forest-Like Shape of ABoxes

To ensure termination is such cases, tableau algorithms employ *blocking* [23], which is based on an important observation about the shape of ABoxes that can be derived from some input ABox $\mathcal{A}$. The individuals in $\mathcal{A}$ are called *named* (shown as black dots), and they can be connected by role assertions in an arbitrary way. The individuals introduced by the $\exists$- and $\geq$-rules are called *blockable* (shown as white dots). For example, if $\exists R.C(a)$ is expanded into $R(a, s)$ and $C(s)$, then s is called a blockable individual and it is an *R-successor* of a. It is not difficult to see that no tableau inference rule can connect s with some element of $\mathcal{A}$: the individual s can participate only in inferences that derive an assertion of the form $D(s)$, create a new successor of s, or, in the presence of (local) reflexivity, connect s to itself. Hence, each ABox $\mathcal{A}'$ obtained from $\mathcal{A}$ can be seen as a "forest" of the form shown in Figure 3: each named individuals can be arbitrarily connected to other named individuals and to a tree of blockable successors. The *concept label* $\mathcal{L}_{\mathcal{A}}(s)$ is defined as the set of all concepts C such that $C(s) \in \mathcal{A}$, and the *edge label* $\mathcal{L}_{\mathcal{A}}(s, s')$ as the set of all atomic roles such that $R(s, s') \in \mathcal{A}$.

The forest-like structure of ABoxes enables blocking. DLs such as $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ allow for inverse roles and number restrictions, which requires *ancestor pairwise blocking* [23]: for s, s', t, and t' individuals of an ABox $\mathcal{A}$ as shown in Figure 3, t blocks s (shown by a double edge on s) if and only if $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$, $\mathcal{L}_{\mathcal{A}}(s') = \mathcal{L}_{\mathcal{A}}(t')$, $\mathcal{L}_{\mathcal{A}}(s, s') = \mathcal{L}_{\mathcal{A}}(t, t')$, and $\mathcal{L}_{\mathcal{A}}(s', s) = \mathcal{L}_{\mathcal{A}}(t', t)$; in the presence of (local) reflexivity, we must additionally require that $\mathcal{L}_{\mathcal{A}}(s, s) = \mathcal{L}_{\mathcal{A}}(t, t)$. In tableau algorithms, the $\exists$- and $\geq$-rules are applicable only to nonblocked individuals, which ensures termination: the number of different concept and edge labels is exponential in $|\mathcal{K}|$, so an exponentially long branch in a forest-like ABox must contain a blocked individual, thus limiting the length of each branch in an ABox. Let $\mathcal{A}$ be an ABox as in Figure 3 to which no tableau inference rule is applicable, and in which s is blocked by t. We can construct a model from $\mathcal{A}$ by *unraveling*—that is, by replicating the fragment between s and t infinitely often. Intuitively, blocking ensures that the part of the ABox between s and s' "behaves" just like the part between t and t', so unraveling indeed generates a model. If our logic were able to connect blockable individuals in a non-tree-like way, then unraveling would not generate a model; in fact, the notion of ancestors, descendants, and blocking would itself be ill-defined.

Consider now an "unlucky" run of a tableau algorithm with ancestor pairwise blocking on $\mathcal{K}_2$. The number of elements in Π is exponential, so it can happen that blocking comes into effect only after the algorithm constructs an exponentially deep tree; since the tree is binary, it is doubly exponential in total. In a "lucky" run, the algorithm can always pick B_j instead of C_j; then, the algorithm constructs a polynomially deep binary tree, so the tree is exponential in total. Thus, the and-branching caused by the $\exists$- and $\geq$-rules limits the applicability of tableau algorithms in practice.

3.2 Hypertableau Algorithm at a Glance

In this section we present an informal overview of our hypertableau algorithm that addresses the problems due to or- and and-branching outlined in Section 3.1. We present the algorithm formally in Section 4.

3.2.1 Inference Rules

The hyperresolution [34] calculus has often been used for first-order theorem proving. It works on *clauses*—implications of the form $\bigwedge_{i=1}^{n} U_i \rightarrow \bigvee_{j=1}^{m} V_j$ where U_i and V_j are first-order atoms. The set of atoms $\{U_i\}$ is called the *antecedent*, and $\{V_j\}$ is called the *consequent*; we omit $\rightarrow$ in clauses with the empty antecedent. The hyperresolution inference rule is defined as

$$\frac{A_1 \vee \mathbf{D_1} \qquad \ldots \qquad A_m \vee \mathbf{D_m} \qquad B_1 \wedge \ldots \wedge B_m \rightarrow C_1 \vee \ldots \vee C_k}{\mathbf{D_1}\sigma \vee \ldots \vee \mathbf{D_m}\sigma \vee C_1\sigma \vee \ldots \vee C_k\sigma}$$

where $\mathbf{D_i}$ is a possibly empty disjunction of literals and σ is the most general unifier of $(A_1, B_1), \ldots, (A_m, B_m)$.[3]

The hypertableau calculus [6] is based on the observation that, if the literals in a clause $C_1 \vee \ldots \vee C_n$ do not share variables, we can replace the clause with a nondeterministically chosen atom C_i that we assume to be true. A hypertableau inference thus combines hyperresolution with splitting from the DPLL calculus for the propositional logic [9] and it can be written as

$$\frac{A_1 \quad \ldots \quad A_m \qquad B_1 \wedge \ldots \wedge B_m \rightarrow C_1 \vee \ldots \vee C_k}{C_1\sigma \quad | \quad \ldots \quad | \quad C_k\sigma}$$

where σ is the most general unifier of $(A_1, B_1), \ldots, (A_m, B_m)$ and $|$ represents or-branching. On Horn clauses, the hypertableau calculus behaves deterministically; thus, the calculus exhibits a "minimal" amount of nondeterminism in general.

The hypertableau calculus from [6] can be easily applied to DLs: GCIs can be translated into first-order formulae [7], which can then be converted into clauses,

[3] As it is standard in resolution theorem proving, the notation $A_i \vee \mathbf{D_i}$ does not imply that A_i is the left-most disjunct in the disjunction.

as shown in the following example.

$$A \sqsubseteq \exists R.B \;\rightsquigarrow\; \forall x : [A(x) \rightarrow \exists y : R(x,y) \wedge B(y)] \;\rightsquigarrow\; \begin{array}{c} A(x) \rightarrow B(f(x)) \\ A(x) \rightarrow R(x, f(x)) \end{array}$$

Let $\mathcal{A}$ be an ABox containing the assertions $A(a)$, $R(a,b)$, and $B(b)$. The GCI $A \sqsubseteq \exists R.B$ is clearly satisfied in $\mathcal{A}$, so there is no need to perform any inference. The clauses obtained by skolemization, however, are not satisfied in $\mathcal{A}$, so the hypertableau calculus derives $R(a, f(a))$ and $B(f(a))$. Hence, skolemization may make the calculus perform unnecessary inferences, which may be inefficient.

Therefore, instead of working on skolemized clauses, our calculus first preprocesses a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$ into a pair $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$, where $\Xi_{\mathcal{A}}(\mathcal{K})$ is an ABox and $\Xi_{\mathcal{TR}}(\mathcal{K})$ is a set of *DL-clauses*—implications of the form $\bigwedge_{i=1}^{n} U_i \rightarrow \bigvee_{j=1}^{m} V_j$, 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 $x \approx y$. The preprocessing step is introduced formally in Section 4.1. The DL-clauses in $\Xi_{\mathcal{TR}}(\mathcal{K})$ are used in the *Hyp*-rule, which is inspired by the hypertableau inference rule. For example, a GCI $\exists R.\neg A \sqsubseteq B$ is translated into a DL-clause $R(x,y) \rightarrow B(x) \vee A(y)$; then, if an ABox contains $R(a,b)$, the *Hyp*-rule derives either $B(a)$ or $A(b)$.

Apart from the *Hyp*-rule, the calculus contains the $\geq$-rule from the standard tableau calculus to deal with the existential quantifiers. Furthermore, the *Hyp*-rule can derive equalities of the form $s \approx t$, which are dealt with using the $\approx$-rule: whenever $s \approx t \in \mathcal{A}$ and $s \neq t$, the $\approx$-rule replaces s with t or vice versa in all assertions in $\mathcal{A}$; this is usually called *merging*. The $\perp$-rule detects obvious contradictions, which can be of the form $s \not\approx s$, or $A(s)$ and $\neg A(s)$. Finally, the *NI*-rule ensures termination in the presence of nominals, number restrictions, and inverse roles; we discuss this rule in more detail in Section 3.2.4.

Our calculus is related to the tableau calculus for first-order logic presented in [8], which also introduces new constants to satisfy the existential quantifiers in a way that makes it complete for finite satisfiability. $\mathcal{SHOIQ}^+$ and similar DLs, however, do not enjoy the finite model property [3], so the calculus from [8] does provide a decision procedure. Furthermore, the calculus by itself is unlikely to be practicable due to a high degree of nondeterminism.

Hyperresolution has been used to decide several description and modal logics [15,26]. These approaches, however, rely on skolemization, which can be inefficient in practice. Furthermore, these approaches deal with logics that are much weaker than $\mathcal{SHOIQ}^+$; in particular, we are not aware of a hyperresolution-based decision procedure that can handle inverse roles, number restrictions, and nominals.

3.2.2 Anywhere Pairwise Blocking

We employ pairwise blocking from Section 3.1.2 to ensure termination of the calculus; to curb and-branching, however, we extend it to *anywhere pairwise blocking.* The key idea is to extend the set of potential blockers for s beyond the ancestors of s. In doing so, however, we must avoid cyclic blocks: if s is allowed to block t and t can block s, then neither s nor t is guaranteed to have all the successors con-

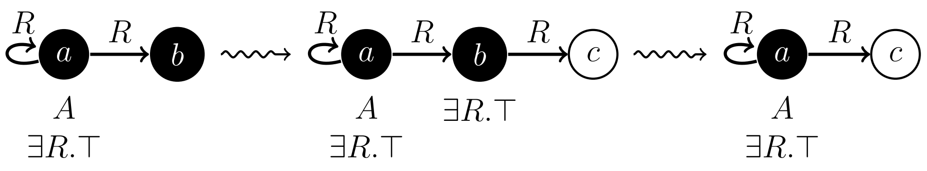


Fig. 4. A Yo-Yo Example

structed, which would render the calculus incomplete. Therefore, we parameterize our algorithm with a strict ordering $\prec$ on individuals that contains the ancestor relation. We allow t to block s only if, in addition to conditions mentioned in Section 3.1.2, we have $t \prec s$. If $\prec$ coincides with the ancestor relation, anywhere blocking becomes equivalent to the ancestor blocking.

Anywhere blocking can reduce and-branching in practice. Consider again the knowledge base $\mathcal{K}_2$ from Section 3.1.2. After we exhaust the exponentially many members of Π, all subsequently created individuals will be blocked. In the best case, we can always choose B_j instead of C_j, so we create a polynomial path in the tree and then use the individuals from that path to block their siblings, as shown in Figure 2b. Hence, with anywhere blocking satisfiability of $\mathcal{K}_2$ can be checked in polynomial time.

3.2.3 Problems Due to Merging

Merging can easily lead to termination problems even for very simple DLs, as we demonstrate using the following example; for simplicity, we present the TBox as a set of DL-clauses $\mathcal{C}_3$.

$$(3)\quad \begin{aligned} \mathcal{A}_3 &= \{\ A(a),\quad \exists R.\top(a),\quad R(a,b),\quad R(a,a)\ \} \\ \mathcal{C}_3 &= \{\ R(x,y_1) \wedge R(x,y_2) \rightarrow y_1 \approx y_2, \qquad A(x) \wedge R(x,y) \rightarrow \exists R.\top(y)\ \} \end{aligned}$$

Consider now the following derivation of the hypertableau calculus on $\mathcal{A}_3$ and $\mathcal{C}_3$, shown in Figure 4: by the second DL-clause, the *Hyp*-rule derives $\exists R.\top(b)$, which the $\exists$-rule expands to $R(b,c)$; then, by the first DL-clause, the *Hyp*-rule derives $b \approx a$, so the $\approx$-rule merges b into a. Clearly, the resulting ABox is isomorphic to the original one (that c is a blockable and b a root individual is not relevant here), so we can repeat the same sequence of inferences, which leads to nontermination. This problem, colloquially known as a “yo-yo,” has, to the best of our knowledge, first been identified in [5].

This problem arises because, due to merging, a can have an unbounded number of blockable R-successors: the blockable individual c is created as an R-successor of b; however, merging b into a makes c a blockable R-successor of a. This, in turn, allows us to apply the DL-clauses from $\mathcal{C}_3$ by mapping x to a an arbitrary number of times, which leads to nontermination.

This problem can be solved by always merging a descendant s into its ancestor t, and *pruning* s before merging—that is, by removing all assertions containing a blockable descendant of s and thus ensuring that t does not “inherit” new succes-

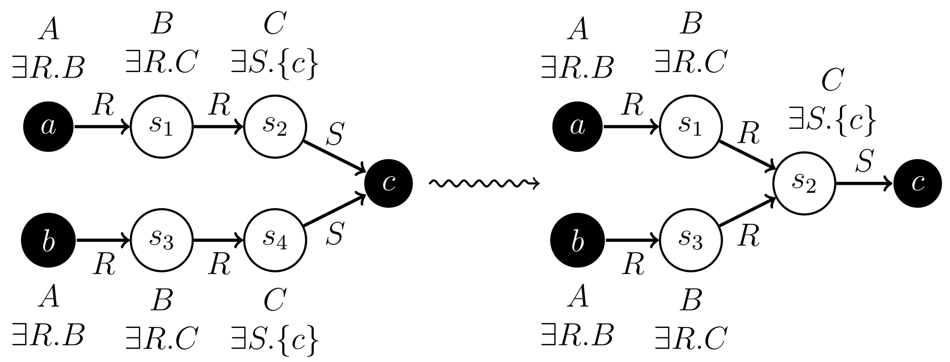


Fig. 5. Non-Tree-Like Structures due to Merging

sors.[4] In our example, before merging b into a, we prune b—that is, we remove the assertion $R(b, c)$. Merging then produces an ABox that represents a model of $\mathcal{A}_3$ and $\mathcal{C}_3$, so the algorithm terminates. Note that pruning is well-defined only because our ABoxes are forest-shaped, cf. Figure 3: if connections between individuals were arbitrary and, in particular, cyclic, it would not be clear which part of the ABox should be pruned.

3.2.4 Nominals

Nominals can make ABoxes non-forest-like, as the following simple example demonstrates. For presentation purposes, we use the concept $\exists R.\{c\}$ in the DL-clauses even though such concepts would be further broken down in our algorithm.

$$\begin{aligned} &\mathcal{A}_4 = \{\ A(a),\quad A(b)\ \} \\ &\mathcal{C}_4 = \{\ A(x) \rightarrow (\exists R.B)(x),\ \ B(x) \rightarrow (\exists R.C)(x),\ \ C(x) \rightarrow (\exists S.\{c\})(x)\ \} \end{aligned} \tag{4}$$

Successive applications of the *Hyp*- and $\exists$-rules on $\mathcal{A}_4$ and $\mathcal{C}_4$ can produce the ABox $\mathcal{A}_4^1$ shown on the left-hand side of Figure 5. This ABox is clearly not forest-shaped: the two role-paths in $\mathcal{A}_4^1$ start at the named individuals a and b, and end in a named individual c. If role relations between blockable individuals remain forest-like, however, termination of the derivation can be ensured easily. Some DLs that include nominals produce only such *extended forest-like* ABoxes, but the property is lost if the DL also includes both inverse roles and number restrictions [20].

Assume now that we extend $\mathcal{C}_4$ with the DL-clause $S(y_1, x) \wedge S(y_2, x) \rightarrow y_1 \approx y_2$ (which axiomatizes S to be inverse-functional). On $\mathcal{A}_4^1$, the *Hyp*-rule then derives $s_2 \approx s_4$. Note that both s_2 and s_4 are blockable individuals; furthermore, neither individual is an ancestor of the other, so we can merge, say, s_4 into s_2. This produces the ABox $\mathcal{A}_4^2$ shown on the right-hand side of Figure 5, and the assertion $R(s_3, s_2)$ makes $\mathcal{A}_4^2$ non-forest-shaped. By extending the example, it is possible to use nominals, inverse roles, and number restrictions to arrange blockable individ-

[4] In [23], the successors are not physically removed, but are marked as "not present" by setting their edge labels to $\emptyset$. This has exactly the same effect as pruning.

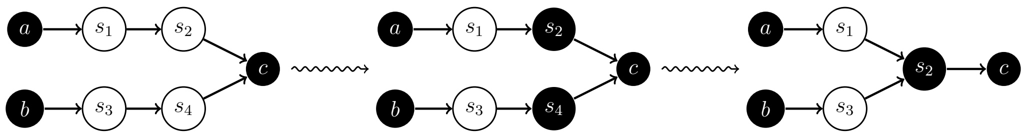


Fig. 6. The Introduction of Root Individuals

uals in cycles. This loss of forest-shaped models prevents us from using blocking and pruning, and thus invalidates our termination arguments.

To solve this problem, we need to extend the arbitrarily interconnected part of $\mathcal{A}_4^2$, cf. Figure 3: we can change the status of s_2 from a blockable into a *root individual*—that is, an individual similar to the named ones in that it can be arbitrarily interconnected. After this change, only s_1 and s_3 are blockable in $\mathcal{A}_4^2$, so the ABox has the extended forest-like shape and we can apply blocking and pruning as usual. This is schematically shown in Figure 6. More generally, we apply the following test (*): if an ABox $\mathcal{A}$ contains assertions $R(s,a)$ and $A(s)$ where a is a root nor a named individual, and s is a blockable individual that is not a successor of a, and if a must satisfy an at-most restriction of the form $\leq n\, R^-.A$, then we change s into a root individual.

This solution, however, introduces another problem: the number of root individuals can now grow arbitrarily, as shown in the following example.

$$(5)\qquad \begin{aligned} \mathcal{A}_5 &= \{\ A(b)\ \} \\ \mathcal{C}_5 &= \left\{ \begin{array}{ll} A(x) \rightarrow (\exists R.A)(x), & A(x) \rightarrow (\exists S.\{a\})(x), \\ S(y_1,x) \wedge S(y_2,x) \wedge S(y_3,x) \rightarrow y_1 \approx y_2 \vee y_2 \approx y_3 \vee y_1 \approx y_3 & \end{array} \right\} \end{aligned}$$

On $\mathcal{A}_5$ and $\mathcal{C}_5$, our calculus can produce the ABox $\mathcal{A}_5^1$ shown on the left-hand side of Figure 7. Note that c and d satisfy condition (*), so we change them into root individuals. Furthermore, the third DL-clause from $\mathcal{C}_5$ is not satisfied, so the *Hyp*-rule derives $c \approx b$, and the $\approx$-rule can merge c into b. Since c is now not a blockable individual, we cannot prune it, so we obtain the ABox $\mathcal{A}_5^2$ shown in the middle of Figure 7.[5] Since $\exists R.A(d)$ is not satisfied, we can extend $\mathcal{A}_5^2$ with $R(d,e)$, $A(e)$, $\exists R.A(e)$, $\exists S.\{a\}(e)$, and $S(e,a)$; however, this makes the third DL-clause from $\mathcal{C}_5$ not satisfied, so we can merge d into b. This produces the ABox $\mathcal{A}_5^3$ that is isomorphic to $\mathcal{A}_5^1$, so we can repeat the same inferences forever.

We solve this problem by introducing a new *NI-rule*, which refines condition (*). Assume that an ABox $\mathcal{A}$ contains assertions $R(s,a)$ and $A(s)$ where s is a blockable individual that is not a successor of a root or a named individual a; furthermore, assume that a must satisfy an at-most restriction of the form $\leq n\, R^-.A$. In any model of $\mathcal{A}$, we can then have at most n different individuals b_i that participate in assertions of the form $R(b_i,a)$ and $A(b_i)$. Hence, we associate with a in advance a set of n fresh root individuals $\{b_1,\ldots,b_n\}$ to serve as R^--neighbors for a. Then,

[5] To reduce clutter, we do not repeat the labels of individuals.

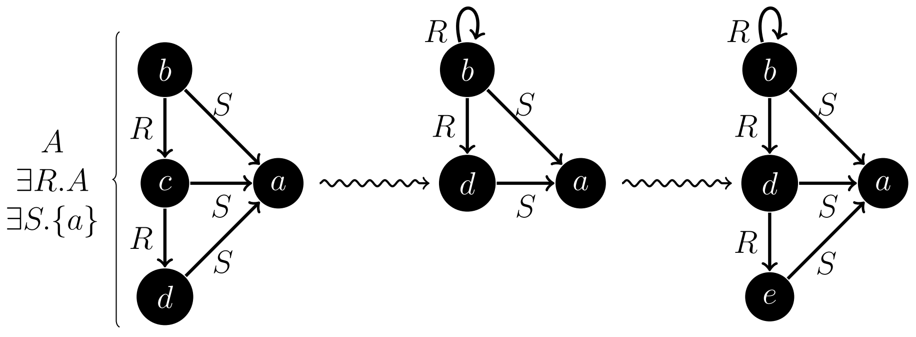


Fig. 7. A Yo-Yo With Root Individuals

to turn s into a root individual, we nondeterministically choose b_j from this set and merge s into b_j. In this way, the number of new root individuals that can be introduced on the individual a is limited to n.

When formulating the *NI*-rule, we are faced with a technical problem: concepts of the form $\leq n\, R.A$ are translated in our calculus into DL-clauses, which makes testing the condition from the previous paragraph difficult. For example, an application of the *Hyp*-rule to the third DL-clause in (5) (obtained from the axiom $\top \sqsubseteq\, \leq 2\, S^-.\top$) can produce an equality such as $c \approx b$. This equality alone does not reflect the fact that a must satisfy the at-most restriction $\leq 2\, S^-.\top$. To enable the application of the *NI*-rule, we introduce a notion of *annotated equalities*, in which the annotations establish an association with the at-most restriction. The third DL-clause from (5) is thus represented in our algorithm using as follows:

$$\begin{aligned} (6) \qquad & S(y_1,x) \wedge S(y_2,x) \wedge S(y_3,x) \rightarrow \\ & \qquad y_1 \approx y_2 \, @^x_{\leq 1\, S^-.\top} \vee y_2 \approx y_3 \, @^x_{\leq 1\, S^-.\top} \vee y_1 \approx y_3 \, @^x_{\leq 1\, S^-.\top} \end{aligned}$$

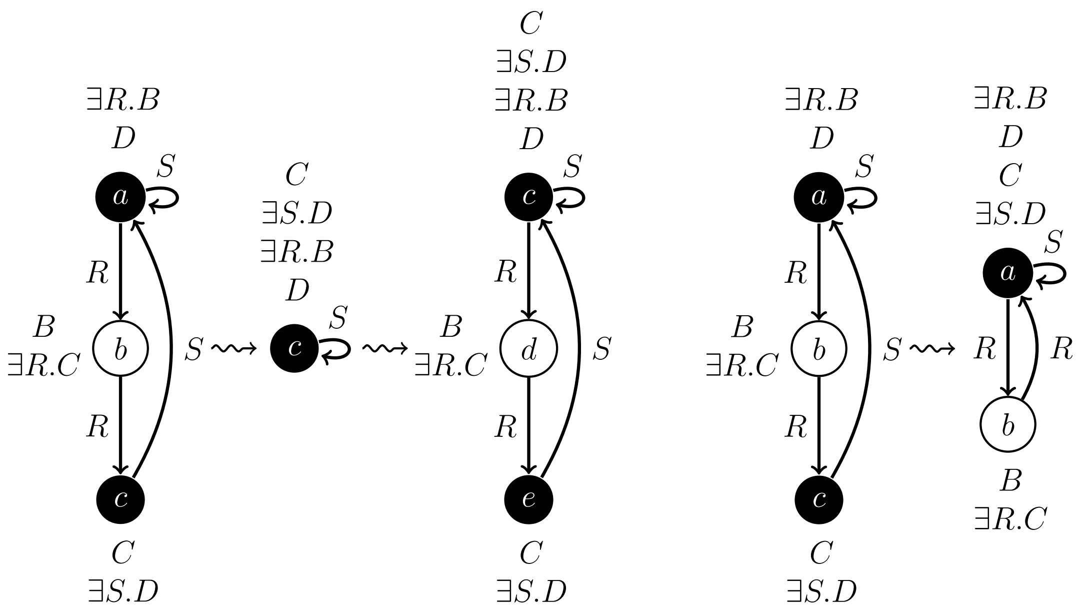
The *Hyp*-rule then derives $c \approx b \, @^a_{\leq 1\, S^-.\top}$, which has the same meaning as $c \approx b$; however, the annotation says that both b and c must also be merged with one of the (two) individuals reserved for use as an S^--neighbor of a.

3.2.5 Nominals and Merging

Repeated merging between root individuals can lead to nontermination. Consider an application of the hypertableau calculus to the following knowledge base:

$$\begin{aligned} (7) \qquad \mathcal{A}_6 &= \left\{ \; S(a,a), \quad \exists R.B(a) \; \right\} \\ \mathcal{C}_6 &= \left\{ \begin{array}{ll} B(x) \rightarrow \exists R.C(x), & C(x) \rightarrow \exists S.D(x), \\ D(x) \rightarrow x \approx a, & S(y_1,x) \wedge S(y_2,x) \rightarrow y_1 \approx y_2 \end{array} \right\} \end{aligned}$$

At first, the calculus introduces two new blockable individuals, after which the *NI*-rule converts the last one into a root individual; the resulting ABox $\mathcal{A}_6^1$ is shown in the left-hand side of Figure 8a. Since S is inverse-functional, the individuals a and c must be merged. Neither individual is an ancestor of the other, so we can choose to merge a into c. The blockable individual b is then pruned, and the resulting ABox is shown in the middle part of Figure 8a. The existential restriction $\exists R.B$ on



(a) Nonterminating Variant (b) Terminating Variant

Fig. 8. The "Caterpillar" Example

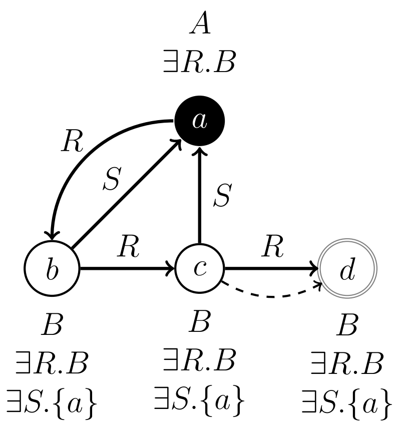
c, however, is not satisfied, so a similar sequence of rule applications constructs the ABox $\mathcal{A}_6^2$ shown in the right-hand side of of Figure 8a. This ABox is isomorphic to $\mathcal{A}_6^1$, so the same inferences can be repeated forever.

This problem can be intuitively explained by the following observation. The *NI*-rule introduces fresh root individuals as neighbors of an existing root individual; thus, each root individual in an ABox can be seen as a part of a "chain" showing which individual caused the introduction of each root individual. Each chain is initially anchored at a named individual: such individuals occur in the input ABox and are not introduced by the *NI*-rule. The length of a path of blockable individuals can be used to limit the length of the "chains" of root individuals. If we allow chain anchors to be removed from an ABox, then the chains remain limited in length in any given ABox; however, over the course of derivation, one end of the chain can be extended indefinitely as the other end is shortened.

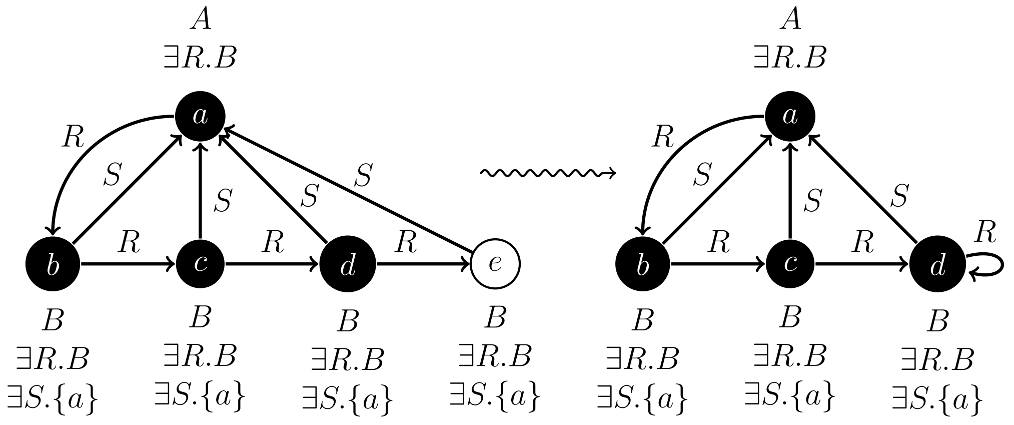
We solve this problem by allowing named individuals to be merged only into other named individuals. This ensures that each chain of root individuals always remains anchored at a named individual, which effectively limits the distance of each root individual to the maximal length of the chain. In our example, instead of merging a into c, we merge c into a, which results in the ABox shown in Figure 8b. No derivation rule is applicable to this ABox, so the algorithm terminates.

3.2.6 The NI-Rule and Unraveling

The *NI*-rule is required not only to ensure that ABoxes are forest shaped, but also to enable the application of blocking and unraveling. Consider the following example knowledge base. Intuitively, the axioms of the knowledge base say that



(a) Premature Blocking



(b) A Correct Derivation

Fig. 9. The *NI*-rule and Unraveling

the individual a can have no R^--neighbors, and that there is an infinite chain of individuals each of is an S^--neighbor of a.

$$\mathcal{A}_7 = \{\ A(a),\ (\exists R.B)(a),\ \}$$

$$\mathcal{C}_7 = \left\{ \begin{array}{l} A(x) \wedge R(y,x) \rightarrow \bot,\quad B(x) \rightarrow (\exists R.B)(x),\quad B(x) \rightarrow (\exists S.\{a\})(x), \\ R(y_1,x) \wedge R(y_2,x) \rightarrow y_1 \approx y_2, \\ S(y_1,x) \wedge S(y_2,x) \wedge S(y_3,x) \wedge S(y_4,x) \rightarrow \\ \qquad y_1 \approx y_2 \vee y_1 \approx y_3 \vee y_1 \approx y_4 \vee y_2 \approx y_3 \vee y_2 \approx y_4 \vee y_3 \approx y_4, \end{array} \right\} \tag{8}$$

Without the *NI*-rule, an application of our calculus to $\mathcal{A}_7$ and $\mathcal{C}_7$ might produce the ABox $\mathcal{A}_7^1$ shown in Figure 9. The individual d is blocked in $\mathcal{A}_7^1$ by the individual c, so the derivation terminates. Note that the last DL-clause from $\mathcal{C}_7$ (which corresponds to the axiom $\top \sqsubseteq\ \leq 3\, S^-$) is satisfied: a is the only individual in $\mathcal{A}_7^1$ that has S^--neighbors and it has only two such neighbors. To construct a model from $\mathcal{A}_7^1$, we unravel the blocked parts of the ABox—that is, we construct an infinite path that extends past d by "duplicating" the fragment of the model between c and d an infinite number of times. This, however, creates additional S^--neighbors of a, which invalidates the last DL-clause from $\mathcal{C}_7$; thus, the unraveled ABox does not define a model of $\mathcal{A}_7$ and $\mathcal{C}_7$.

The *NI*-rule elegantly solves this problem. Since a must satisfy an at-most restriction of the form $\leq 3\, S^-$, as soon as $S(b, a)$, $S(c, a)$, and $S(d, a)$ are derived, the *NI*-rule is applied to turn b, c, and d into root individuals. This corrects the problems with unraveling: root individuals do not become blocked, so we introduce another fresh blockable individual e. This individual is merged with another S^--neighbor of a, producing an individual with two R^--neighbors. R is inverse-functional, however, so the neighbors are merged. Merging continues until b has been merged into a, causing a to become its own R-neighbor, at which point our algorithm correctly determines that $\mathcal{A}_7 \cup \mathcal{C}_7$ is inconsistent.

4 The Satisfiability Checking Algorithm

We now present the hypertableau algorithm that can be used to check satisfiability of a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$. Our algorithm consists of two phases: the *preprocessing* phase is the subject of Section 4.1, and the *hypertableau* phase is the subject of Section 4.2.

4.1 Preprocessing

The goal of the preprocessing phase is to transform a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$ into an ABox and a set of *DL-clauses* that are equisatisfiable with $\mathcal{K}$.

Definition 1 (DL-Clause)**.** Let N_V be a set of *variables* disjoint from the set of individuals N_I. An *atom* is an expression of the form $B(s)$, $\geq n\, R.B(s)$, $R(s, t)$, or $s \approx t$, for s and t individuals or variables, B a literal concept, R an atomic role, and n a positive integer. A *DL-clause* is an expression of the form

$$U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n$$

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*. The empty antecedent and the empty consequent of a DL-clause are written as $\top$ and $\bot$, respectively.

Let $I = (\triangle^I, \cdot^I)$ be an interpretation and $\mu : N_V \rightarrow \triangle^I$ a mapping of variables to elements of the interpretation domain. 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 $\mathcal{C}$ in I and μ is defined in Table 2.

4.1.1 Elimination of Transitivity Axioms

Transitivity axioms are handled in the tableau algorithms by the $\forall_+$-rule: if R is transitive and an ABox contains $\forall R.C(s)$ and $R(s, t)$, the $\forall_+$-rule derives $\forall R.C(t)$. In our algorithm, however, concepts of the form $\forall R.C$ are translated

Table 2
Satisfaction of DL-Clauses in an Interpretation

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

into DL-clauses, so the $\forall_+$-rule cannot be applied. Therefore, instead of handling transitivity directly, we encode a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$ into an equisatisfiable $\mathcal{ALCHOIQ}^+$ knowledge base $\Omega(\mathcal{K})$. This encoding eliminates all transitivity axioms, but simulates their effects using additional GCIs.

Definition 2. Given a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$, the *concept closure* of $\mathcal{K}$ is the smallest set of concepts $\mathsf{clos}(\mathcal{K})$ such that

- if $C \sqsubseteq D \in \mathcal{T}$, then $\mathsf{nnf}(\neg C \sqcup D) \in \mathsf{clos}(\mathcal{K})$,
- if $C(a) \in \mathcal{A}$, then $\mathsf{nnf}(C) \in \mathsf{clos}(\mathcal{K})$,
- if $C \in \mathsf{clos}(\mathcal{K})$ and D syntactically occurs in C, then $D \in \mathsf{clos}(\mathcal{K})$,
- if $\leq n\, R.C \in \mathsf{clos}(\mathcal{K})$, then $\dot\neg C \in \mathsf{clos}(\mathcal{K})$, and
- if $\forall R.C \in \mathsf{clos}(\mathcal{K})$, $S \sqsubseteq^*_\mathcal{R} R$, and $\mathsf{Tra}(S) \in \mathcal{R}$, then $\forall S.C \in \mathsf{clos}(\mathcal{K})$.

The Ω-*encoding* of $\mathcal{K}$ is the $\mathcal{ALCHOIQ}^+$ knowledge base $\Omega(\mathcal{K}) = (\mathcal{R}', \mathcal{T}', \mathcal{A})$ where $\mathcal{R}'$ is obtained from $\mathcal{R}$ by removing all transitivity axioms and

$$\mathcal{T}' = \mathcal{T} \cup \{\forall R.C \sqsubseteq \forall S.(\forall S.C) \mid \forall R.C \in \mathsf{clos}(\mathcal{K}),\ S \sqsubseteq^*_\mathcal{R} R, \text{ and } \mathsf{Tra}(S) \in \mathcal{R}\}.$$

Similar encodings are known for various description [38] and modal [35] logics. Note that, in order to guarantee decidability [22], number restrictions are allowed in $\mathcal{SHOIQ}^+$ only on simple roles—that is, on roles not having transitive subroles; for similar reasons, role disjointness, irreflexivity, and asymmetry axioms are allowed only on simple roles as well. Therefore, the proof of the following lemma is quite similar to the proofs in [38,35], so we omit it for the sake of brevity. The full proof for the DL $\mathcal{SHIQ}$ can be found [28, Section 5.2], and its generalization to $\mathcal{SHOIQ}^+$ is straightforward.

Lemma 3 ([28])**.** *A $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$ is satisfiable if and only if $\Omega(\mathcal{K})$ is satisfiable, and $\Omega(\mathcal{K})$ can be computed in time polynomial in $|\mathcal{K}|$.*

After the elimination of transitivity axioms, there is no distinction between simple and complex roles. Hence, in the rest of this paper we assume that all roles are simple unless otherwise stated and, without loss of generality, we treat $\exists R.C$ as a syntactic shortcut for $\geq 1\, R.C$.

4.1.2 Normalization

Before a knowledge base is translated into a set of DL-clauses, it is first brought into a *normalized* form. This is done in order to make all implicit negations explicit, and to ensure that the resulting DL-clauses are compatible with blocking.

To understand the first issue, consider the axiom $\neg A \sqsubseteq \neg(\exists R.\exists R.\exists R.B)$. Due to implicit negations, converting this axiom into DL-clauses is not straightforward. Therefore, we replace this axiom with the following equivalent axiom. This makes all negations explicit, so the result can be easily translated into a DL-clause.

(9) $\top \sqsubseteq A \sqcup \forall R.\forall R.\forall R.\neg B \rightsquigarrow R(x, y_1) \wedge R(y_1, y_2) \wedge R(y_2, y_3) \wedge B(y_3) \rightarrow A(x)$

To understand the second issue, consider the knowledge base $\mathcal{K}_8$, consisting of an ABox $\mathcal{A}_8$ and a TBox that corresponds to the set of DL-clauses $\mathcal{C}_8$.

$$\begin{aligned} \mathcal{A}_8 &= \{\ \neg A(a),\ B(a)\ \} \\ \mathcal{C}_8 &= \{\ R(x, y_1) \wedge R(y_1, y_2) \wedge R(y_2, y_3) \wedge B(y_3) \rightarrow A(x), \\ &\qquad B(x) \rightarrow \exists R.B(x)\ \} \end{aligned} \qquad (10)$$

By applying the rules from Section 3.2, our algorithm constructs on $\mathcal{K}_8$ the ABox shown in Figure 10. As discussed in Section 3.1.2, c is now blocked by b; furthermore, no rule is applicable to the ABox, so the algorithm terminates, leading us to believe that $\mathcal{K}_8$ is satisfiable. The ABox, however, does not represent a model of $\mathcal{K}_8$: if we expand $\exists R.B(c)$ into $R(c, d)$ and $B(d)$, by the first DL-clause in $\mathcal{C}_8$ we can derive the assertion $A(a)$, which contradicts $\neg A(a)$. This problem arises because the antecedent of the first DL-clause in $\mathcal{C}_8$ checks for a path of three R-successors, whereas the pairwise blocking condition ensures only that all paths of length two are fully constructed. Intuitively, the antecedents of each DL-clause should check for paths that "fit" into the fully constructed model fragments. We can ensure this by renaming complex concepts into simpler ones. Thus, we transform the culprit DL-clause into the following ones, which check only for paths of length one.

(11) $\top \sqsubseteq A \sqcup \forall R.\neg Q_1 \quad \rightsquigarrow \quad R(x, y) \wedge Q_1(y) \rightarrow A(x)$

(12) $\top \sqsubseteq Q_1 \sqcup \forall R.\neg Q_2 \quad \rightsquigarrow \quad R(x, y) \wedge Q_2(y) \rightarrow Q_1(x)$

(13) $\top \sqsubseteq Q_2 \sqcup \forall R.\neg B \quad \rightsquigarrow \quad R(x, y) \wedge B(y) \rightarrow Q_2(x)$

The application of these DL-clauses to the ABox shown in Figure 10 would additionally derive $Q_2(a)$, $Q_2(b)$, and $Q_1(a)$, so c would not be blocked. The calculus would then expand $\exists R.B(c)$ and discover a contradiction.

To formalize these ideas, we define a normalized form of DL knowledge bases.

Definition 4 (Normalized Form)**.** 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, $\{a\}$, $\forall R.B$, $\exists R.\mathsf{Self}$, $\neg\exists R.\mathsf{Self}$, $\geq n\, R.B$, or $\leq n\, R.B$, for B a literal concept, R a role, and n a nonnegative integer.

A TBox $\mathcal{T}$ is *normalized* if each GCI in it is normalized. An ABox $\mathcal{A}$ is *normalized* if each concept assertion in $\mathcal{A}$ is of the form $B(a)$ or $\geq n\, R.B(a)$, for B a literal

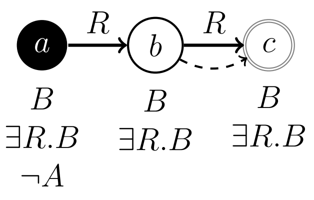


Fig. 10. Incorrect Blocking due to Lack of Normalization

Table 3
The Functions Used in the Normalization

$$\begin{aligned}
\Delta(\mathcal{K}) &= \{\top(a)\} \cup \bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \Delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \Delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2)) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup C') &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \bigcup_{1 \leq i \leq n} \Delta(\top \sqsubseteq \dot{\neg}\alpha_{C'} \sqcup C_i) \\
&\quad \text{for } C' \text{ of the form } C' = C_1 \sqcap \ldots \sqcap C_n \text{ and } n \geq 2 \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \geq n\, R.\alpha_D) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.D) &= \Delta(\top \sqsubseteq \mathbf{C} \sqcup \leq n\, R.\dot{\neg}\alpha_{\dot{\neg}D}) \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_{\dot{\neg}D} \sqcup \dot{\neg}D) \\
\Delta(\top \sqsubseteq \mathbf{C} \sqcup \neg\{s\}) &= \begin{cases} \bot & \text{if } \mathbf{C} \text{ is empty,} \\ \Delta(\mathbf{C}(s)) & \text{otherwise.} \end{cases} \\
\Delta(D(s)) &= \{\alpha_D(s)\} \cup \Delta(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup \mathsf{nnf}(D)) \\
\Delta(R^-(s,t)) &= \{R(t,s)\} \\
\Delta(\beta) &= \{\beta\} \text{ for any other axiom } \beta
\end{aligned}$$

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

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

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 a is a fresh individual. Our notation takes into account that $\sqcap$ and $\sqcup$ are commutative; hence, the concept C' in a disjunction of the form $\mathbf{C} \sqcup C'$ is not necessarily the right-most disjunct.

concept, each role assertion in $\mathcal{A}$ contains only atomic roles, and $\mathcal{A}$ contains at least one assertion. An $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is *normalized* if $\mathcal{T}$ and $\mathcal{A}$ are normalized.

The following transformation can be used to normalize a knowledge base.

Definition 5 (Normalization)**.** For an $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$, the knowledge base $\Delta(\mathcal{K})$ is computed as shown in Table 3.

Normalization can be seen as an optimized variant of the well-known structural transformation [32]. An application of the structural transformation to axiom (9) would replace each complex subconcept with a positive atomic concept, eventually producing the axiom $\top \sqsubseteq A \sqcup \forall R.Q_1$. This axiom cannot be translated into a Horn DL-clause, whereas the original axiom (9) can; thus, the structural transformation can destroy Horn-ness. To prevent this, we introduce the function $\mathsf{pos}(C)$ (c.f. Table 3) that returns false if the clausification of C does not require adding atoms into the consequent of a DL-clause. When replacing an occurrence of a concept C in a concept D, if $\mathsf{pos}(C) = \mathsf{false}$, we replace C in D with a negative literal concept $\neg Q_C$; otherwise, if $\mathsf{pos}(C) = \mathsf{true}$, we replace C in D with a positive literal concept Q_C. If $\mathcal{K}$ is expressed in the DL Horn-$\mathcal{SHIQ}$ [25], this transformation produces only Horn DL-clauses, thus allowing for reasoning without any nondeterminism.

The following lemma captures the important properties of normalization.

Lemma 6. *An $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K}$ is satisfiable if and only if $\Delta(\mathcal{K})$ is satisfiable. Furthermore, $\Delta(\mathcal{K})$ is normalized and it can be computed in time polynomial in $|\mathcal{K}|$.*

PROOF. It is easy to see that our transformation is a syntactic variant of the structural transformation from [32], so $\mathcal{K}$ and $\Delta(\mathcal{K})$ are equisatisfiable. 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 until the GCI becomes normalized. Furthermore, it adds $\top(a)$ to the ABox so that the ABox is not empty, and it replaces all inverse role assertions with equivalent assertions on the atomic roles. Thus, $\Delta(\mathcal{K})$ is normalized. Finally, each occurrence of a concept in $\mathcal{K}$ can be replaced with a new atomic concept at most once, and all necessary syntactic transformations can be performed in polynomial time, so $\Delta(\mathcal{K})$ can be computed in polynomial time. $\square$

4.1.3 Translation into DL-Clauses

We now introduce the notion of HT-clauses—syntactically restricted DL-clauses on which our hypertableau calculus is guaranteed to terminate. In the rest of this paper, we often use the following function ar. Given a role R and variables or constants s and t, this function returns an atom that is semantically equivalent to $R(s,t)$ but that contains an atomic role.

$$\mathsf{ar}(R,s,t) = \begin{cases} R(s,t) & \text{if } R \text{ is an atomic role} \\ S(t,s) & \text{if } R \text{ is an inverse role and } R = S^- \end{cases}$$

Definition 7 (HT-Clause)**.** We assume that the set of atomic concepts N_C contains a *nominal guard concept* O_a for each individual a; these concepts should not occur in any input knowledge base.

An *at-most equality* is an atom of the form $s \approx t \,@^{u}_{\leq n\, R.B}$, where s, t, and u are constants or variables, n is a nonnegative integer, R is a role, and B is a literal concept. Semantically, this atom is equivalent to $s \approx t$; the *annotation* $@^{u}_{\leq n\, R.B}$ is only used to ensure termination of the hypertableau calculus.

An *HT-clause* is a DL-clause r of the form

$$U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n \tag{14}$$

in which it must be possible to separate the variables into a *center variable* x, a set of *branch variables* y_i, and a set of *nominal variables* z_j such that the following properties hold, for A an atomic concept, B a literal but not a nominal guard concept, C a concept of the form B or $\geq n\, T.B$, O_a a nominal guard concept, R and S atomic roles, and T a role.

- Each atom in the antecedent of r is of the form $A(x)$, $R(x,x)$, $R(x,y_i)$, $R(y_i,x)$, $A(y_i)$, or $A(z_j)$.
- Each atom in the consequent of r is of the form $C(x)$, $C(y_i)$, $R(x,x)$, $R(x,y_i)$, $R(y_i,x)$, $R(x,z_j)$, $R(z_j,x)$, $x \approx z_j$, $y_i \approx z_j$, or $y_i \approx y_j \,@^x_{\leq n\, T.B}$.
- Each y_i occurs in the antecedent of r in an atom of the form $R(x,y_i)$ or $R(y_i,x)$.
- Each z_j occurs in the antecedent of r in an atom of the form $O_a(z_j)$.
- Each equality $y_i \approx y_j \,@^x_{\leq h\, T.A}$ in the consequent of r must occur in a subclause of r of the form (15) and no y_k with $1 \leq k \leq h+1$ should occur elsewhere in r.

$$\ldots \bigwedge_{k=1}^{h+1} [\mathsf{ar}(T,x,y_k) \wedge A(y_k)] \ldots \rightarrow \ldots \bigvee_{1 \leq k < \ell \leq h+1} y_k \approx y_\ell \,@^x_{\leq h\, T.A} \cdots \tag{15}$$

- Each equality $y_i \approx y_j \,@^x_{\leq n\, T.\neg A}$ in the consequent of r must occur in a subclause of r of the form (16) and no y_k with $1 \leq k \leq h+1$ should occur elsewhere in r.

$$\ldots \bigwedge_{k=1}^{h+1} \mathsf{ar}(T,x,y_k) \ldots \rightarrow \ldots \bigvee_{k=1}^{h+1} A(y_k) \vee \bigvee_{1 \leq k < \ell \leq h+1} y_k \approx y_\ell \,@^x_{\leq h\, T.\neg A} \cdots \tag{16}$$

The notion of HT-clauses is more general than what is strictly needed to capture $\mathcal{ALCHOIQ}^+$ constructs. For example, the HT-clause $R(x,y) \wedge A(y) \rightarrow S(x,y)$ expresses a form of *relativized* role inclusions; furthermore, HT-clauses of the form $R(x,y) \wedge S(y,x) \rightarrow U(x,y) \vee T(y,x)$ can fully capture *safe* role expressions [38].

We now show how to transform a normalized $\mathcal{ALCHOIQ}^+$ knowledge base into a set of HT-clauses.

Definition 8 (Clausification)**.** The *clausification* of a normalized $\mathcal{ALCHOIQ}^+$ knowledge base $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ is the pair $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$, where $\Xi_{\mathcal{TR}}(\mathcal{K})$ is the set of DL-clauses and $\Xi_{\mathcal{A}}(\mathcal{K})$ is the ABox defined as shown in Table 4.

By Definition 4, normalized knowledge bases do not contain negated nominal concepts of the form $\neg\{a\}$: if present in the input knowledge base, such concepts are converted to ABox assertions during normalization; hence, Table 4 need not handle concepts of the form $\neg\{a\}$. Positive nominal concepts are naturally translated into equalities containing constants; for example, $\top \sqsubseteq \neg A \sqcup \{a\}$ naturally corresponds to the DL-clause $A(x) \rightarrow x \approx a$. Such clauses cause problems for the $\approx$-rule. For

Table 4
Translation of a Normalized Knowledge Base to HT-Clauses

$$\Xi_{\mathcal{T}}(\mathcal{T}) = \{\bigwedge_{i=1}^{n} \mathsf{lhs}(C_i) \rightarrow \bigvee_{i=1}^{n} \mathsf{rhs}(C_i) \mid \text{ for each } \top \sqsubseteq \bigsqcup_{i=1}^{n} C_i \text{ in } \mathcal{T}\}$$

$$\begin{aligned}\Xi_{\mathcal{R}}(\mathcal{R}) = {} & \{\mathsf{ar}(R,x,y) \rightarrow \mathsf{ar}(S,x,y) \mid \text{ for each } R \sqsubseteq S \text{ in } \mathcal{R}\} \cup \\ & \{\mathsf{ar}(S_1,x,y) \wedge \mathsf{ar}(S_2,x,y) \rightarrow \bot \mid \text{ for each } \mathsf{Dis}(S_1,S_2) \in \mathcal{R}\} \cup \\ & \{\top \rightarrow \mathsf{ar}(R,x,x) \mid \text{ for each } \mathsf{Ref}(R) \in \mathcal{R}\} \cup \\ & \{\mathsf{ar}(S,x,x) \rightarrow \bot \mid \text{ for each } \mathsf{Irr}(S) \in \mathcal{R}\} \cup \\ & \{\mathsf{ar}(R,x,y) \rightarrow \mathsf{ar}(R,y,x) \mid \text{ for each } \mathsf{Sym}(R) \in \mathcal{R}\} \cup \\ & \{\mathsf{ar}(S,x,y) \wedge \mathsf{ar}(S,y,x) \rightarrow \bot \mid \text{ for each } \mathsf{Asy}(S) \in \mathcal{R}\}\end{aligned}$$

$$\Xi_{\mathcal{TR}}(\mathcal{K}) = \Xi_{\mathcal{T}}(\mathcal{T}) \cup \Xi_{\mathcal{R}}(\mathcal{R})$$

$$\Xi_{\mathcal{A}}(\mathcal{K}) = \mathcal{A} \cup \{O_a(a) \mid \text{for each } \{a\} \text{ occurring in } \mathcal{K}\}$$

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		$A(x)$
$\neg A$	$A(x)$	
$\{a\}$	$O_a(z_C)$	$x \approx z_C$
$\geq n\, R.A$		$\geq n\, R.A(x)$
$\geq n\, R.\neg A$		$\geq n\, R.\neg A(x)$
$\exists R.\mathsf{Self}$		$\mathsf{ar}(R,x,x)$
$\neg\exists R.\mathsf{Self}$	$\mathsf{ar}(R,x,x)$	
$\forall R.A$	$\mathsf{ar}(R,x,y_C)$	$A(y_C)$
$\forall R.\neg A$	$\mathsf{ar}(R,x,y_C) \wedge A(y_C)$	
$\leq n\, R.A$	$\bigwedge_{i=1}^{n+1} [\mathsf{ar}(R,x,y_C^i) \wedge A(y_C^i)]$	$\bigvee_{1 \leq i < j \leq n+1} y_C^i \approx y_C^j \,@^x_{\leq n\, R.A}$
$\leq n\, R.\neg A$	$\bigwedge_{i=1}^{n+1} \mathsf{ar}(R,x,y_C^i)$	$\bigvee_{i=1}^{n+1} A(y_C^i) \vee \bigvee_{1 \leq i < j \leq n+1} y_C^i \approx y_C^j \,@^x_{\leq n\, R.\neg A}$

Note: Each $y_C^{(i)}$ and z_C is a fresh variable unique for C (and i).

example, given an equality assertion $a \approx b$, the $\approx$-rule replaces all occurrences of a with b. Such replacements should be performed not only in the facts, but in the DL-clauses as well; thus, the mentioned DL-clause should be replaced with $A(x) \rightarrow x \approx b$. This has the drawback that the set of DL-clauses is not fixed in the course of derivation. We solve this problem by "extracting" all constants into the ABox: our clausification algorithm converts $\top \sqsubseteq \neg A \sqcup \{a\}$ into the DL-clause $A(x) \wedge O_a(z_{\{a\}}) \rightarrow x \approx z_{\{a\}}$ and the fact $O_a(a)$. All constants are thus "pushed" into the facts, so the $\approx$-rule can perform replacements only in the facts.

Lemma 9. *Let $\mathcal{K}$ be a normalized $\mathcal{ALCHIQ}$ knowledge base. Then, $\mathcal{K}$ is equisatisfiable with $\Xi(\mathcal{K}) = (\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$, and $\Xi_{\mathcal{TR}}(\mathcal{K})$ contains only HT-clauses.*

PROOF. By inspecting Table 4, it is easy to see that $\Xi_{\mathcal{TR}}(KB)$ contains only HT-clauses. The following equivalences are well known to hold [3] between DL concepts and first-order formulae:

$$\begin{aligned}\forall R.B(x) &\equiv \forall y : \neg R(x,y) \vee B(y)\\ \leq n\, R.B(x) &\equiv \forall y_1,\ldots,y_{n+1} : \bigvee_{i=1}^{n+1}[R(x,y_i) \wedge B(y_i)] \rightarrow \bigvee_{i=1}^{n+1}\bigvee_{j=i+1}^{n+1} y_i \approx y_j\\ \{a\}(x) &\equiv x \approx a\end{aligned}$$

Let $\Xi'_{\mathcal{TR}}(\mathcal{K})$ be the set of DL-clauses defined just like $\Xi_{\mathcal{TR}}(\mathcal{K})$, but with the difference that $\mathsf{lhs}(\{a\}) = \top$ and $\mathsf{rhs}(\{a\}) = x \approx a$. Then, $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is obtained from $\mathcal{K}$ by replacing concepts of the form $\forall R.B$, $\leq n\, R.B$ and $\{a\}$ with the equivalent first-order formulae, so $\mathcal{K}$ and $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ are clearly equisatisfiable.

We now show that $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ is equisatisfiable with $(\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$. For the ($\Rightarrow$) direction, each model I' of $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ can be extended to a model I of $(\Xi_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$ by setting $O_a^I = \{a^{I'}\}$ for each nominal guard concept O_a. For the ($\Leftarrow$) direction, it is easy to see that each model I of $\Xi(\mathcal{K})$ is a model of $(\Xi'_{\mathcal{TR}}(\mathcal{K}), \Xi_{\mathcal{A}}(\mathcal{K}))$. In particular, for each HT-clause $\gamma \in \Xi'_{\mathcal{TR}}(\mathcal{K})$, we have $\delta \in \Xi_{\mathcal{TR}}(\mathcal{K})$, where γ and δ are of the form shown below.

$$\begin{aligned}\gamma &= \bigwedge U_i \rightarrow \bigvee V_j \vee \bigvee_{k=1}^{n} x_k \approx a_k \qquad\qquad \rightsquigarrow\\ \delta &= \bigwedge U_i \wedge \bigwedge_{k=1}^{n} O_{a_k}(z_{\{a_k\}}) \rightarrow \bigvee V_j \vee \bigvee_{k=1}^{n} x_k \approx z_{\{a_k\}}\end{aligned}$$

Now if the disjunction $\bigvee_{k=1}^{n} x_k \approx a_k$ in some γ were not true in I for some values of x_k, then clearly δ would not be true in I for the same values of x_k. $\square$

4.2 The Hypertableau Calculus for HT-Clauses

We now present the hypertableau calculus for deciding satisfiability of an ABox $\mathcal{A}$ and a set of HT-clauses $\mathcal{C}$. In our algorithm, we call the individuals that occur in the input ABox *named*. Furthermore, for a named individual a, the *NI*-rule might need to introduce individuals that are unique for a, a role R, a literal concept B, and some integer i; we represent such individuals as $a.\langle R, B, i\rangle$. Since the *NI*-rule might be applied to these individuals as well, we formalize the notion of *root individuals* as finite strings of the form $a.\gamma_1.\ldots.\gamma_n$ where a is a named individual and each γ_ℓ is of the form $\langle R.B.i\rangle$.

In standard tableau algorithms, the tree structure of the model is encoded in the edges between individuals, which always point from parents to children. In contrast, our algorithm encodes the parent-child relationships into individuals themselves: it represents individuals as finite strings of the form $s.i_1, i_2, \ldots, i_n$, where s is a root individual and i_j are integers. For example, $a.2$ is the second child of the named individual a. Individuals with $n \geq 1$ are called *blockable.*

Definition 10 (Hypertableau Algorithm)**.**

Individuals. Given a set of *named* individuals N_I, the set of *root individuals* N_O is the smallest set such that $N_I \subseteq N_O$ and, if $x \in N_O$, then $x.\langle R, B, i\rangle \in N_O$ for each role R, literal concept B, and integer i. The set of *all individuals* N_A is the smallest set such that $N_O \subseteq N_A$ and, if $x \in N_A$, then $x.i \in N_A$ for each integer i. The individuals in $N_A \setminus N_O$ are *blockable individuals.* A blockable 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.

ABoxes. The hypertableau algorithm operates on *generalized ABoxes*, which are obtained by extending the standard notion of ABoxes as follows.

- In addition to standard assertions, an ABox can contain at-most equalities and a special assertion $\perp$ that is false in all interpretations. Furthermore, assertions can refer to the individuals from N_A and not only from N_I.
- Each (in)equality $s \approx t$ ($s \not\approx t$) also stands for the symmetric (in)equality $t \approx s$ ($t \not\approx s$). The same is true for annotated at-most equalities.
- An ABox $\mathcal{A}$ can contain *renamings* of the form $a \mapsto b$ where a and b are root individuals. The relation $\mapsto$ in $\mathcal{A}$ must be acyclic, $\mathcal{A}$ can contain at most one renaming $a \mapsto b$ for an individual a, and, if $\mathcal{A}$ contains $a \mapsto b$, then a should not occur in any assertion or (in)equality in $\mathcal{A}$. An individual b is the *canonical name* of a root individual a in $\mathcal{A}$, written $b = \|a\|_{\mathcal{A}}$, iff $a \mapsto^* b$ and there exists no individual $c \neq b$ such that $b \mapsto^* c$, where $\mapsto^*$ is the reflexive-transitive closure of $\mapsto$ in $\mathcal{A}$.

An ABox that contains only named individuals and no at-most equalities is called an *input ABox.*

Pairwise Anywhere Blocking. For A an atomic concept and R a role, concepts of the form A, $\geq n\, R.A$, or $\geq n\, R.\neg A$ are called *blocking-relevant.* The *labels of an individual* s and *of an individual pair* $\langle s, t\rangle$ in an ABox $\mathcal{A}$ are defined as follows:

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

Let $\prec$ be a strict ordering (i.e., a transitive and irreflexive relation) on N_A 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:

- a blockable individual s is *directly blocked by* a blockable individual t iff, for s' and t' the predecessors of s and t, respectively, we have
  - · t is not blocked,
  - · $t \prec s$,
  - · $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$, $\mathcal{L}_{\mathcal{A}}(s') = \mathcal{L}_{\mathcal{A}}(t')$, and $\mathcal{L}_{\mathcal{A}}(s,s) = \mathcal{L}_{\mathcal{A}}(t,t)$, and
  - · $\mathcal{L}_{\mathcal{A}}(s,s') = \mathcal{L}_{\mathcal{A}}(t,t')$ and $\mathcal{L}_{\mathcal{A}}(s',s) = \mathcal{L}_{\mathcal{A}}(t',t)$;
- s is *indirectly blocked* iff it has a predecessor that is blocked; and
- 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 containing a descendent of s.

Merging. The ABox $\mathsf{merge}_{\mathcal{A}}(s \rightarrow t)$ is obtained from $\mathsf{prune}_{\mathcal{A}}(s)$ by replacing the individual s with the individual t in all assertions and their annotations (but not in renamings) and, if both s and t are root individuals, adding the renaming $s \mapsto t$.

Derivation Rules. Table 5 specifies *derivation rules* that, given an ABox $\mathcal{A}$ and a set of HT-clauses $\mathcal{C}$, derive the ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$. In the *Hyp*-rule, σ is a mapping from the set of variables N_V to the individuals occurring in the assertions of $\mathcal{A}$, and $\sigma(U)$ is the result of replacing each variable x in the atom U with $\sigma(x)$.

Rule Precedence. The $\approx$-rule can be applied to a (possibly annotated) equality $s \approx t$ in an ABox $\mathcal{A}$ only if $\mathcal{A}$ does not contain an equality $s \approx t \,@^{u}_{\leq n\, R.B}$ to which the *NI*-rule is applicable.

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

Derivation. For a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$, 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, for each node $t \in T$, the following properties hold:

- if t the root of T, then $\lambda(t) = \mathcal{A}$;
- if $\bot \in \lambda(t)$ or no derivation rule is applicable to $\lambda(t)$ and $\mathcal{C}$, then t is a leaf of T;
- otherwise, t has children $t_1, \ldots, t_n$ such that $\lambda(t_1), \ldots, \lambda(t_n)$ are exactly the results of applying one (arbitrarily chosen, but respecting the rule precedence) applicable rule to $\lambda(t)$ and $\mathcal{C}$.

We stress several important aspects of Definition 10. If the preconditions of the *NI*-rule are satisfied for an at-most equality $s \approx t \,@^{u}_{\leq n\, R.B}$, then the rule must be applied even if $s = t$; hence, such an equality plays a role in a derivation even though it is a logical tautology. Furthermore, if u is not a root individual, then the *NI*-rule is never applied to such an at-most equality, so the equality can be eagerly simplified into $s \approx t$. Finally, if $\mathcal{C}$ has been obtained by a translation of a DL knowledge base that does not use nominals, inverse roles, or number restrictions, then the precondition of the *NI*-rule will never be satisfied, so we need not keep track of annotations at all.

We next introduce HT-ABoxes, which capture the notion of a forest-shaped interpretation.

Definition 11 (HT-ABoxes)**.** An ABox $\mathcal{A}$ is an *HT-ABox* if all of its assertions satisfy the following conditions, where R is an atomic role, $s, t \in N_A$, $a \in N_O$, and i and j are integers.

(1) Each role assertion in $\mathcal{A}$ is of the form $R(a, s)$, $R(s, a)$, $R(s, s.i)$, $R(s.i, s)$, or $R(s, s)$.

(2) Each equality in $\mathcal{A}$ either is of the form $s \approx t \,@^{a}_{\leq n\, R.B}$ with s a blockable individual that is not a successor of a and t a blockable individual, or it is a possibly annotated equality of the form $s.i \approx s.j$, $s.i \approx s$, $s.i.j \approx s$, $s \approx s$, or $s \approx a$. (The symmetry of $\approx$ applies in all these cases as usual.)

(3) If $\mathcal{A}$ contains $O_a(s)$ for O_a a nominal guard concept, then $s \in N_I$.

Table 5
Derivation Rules of the Tableau Calculus

Hyp-rule	If 1. $U_1 \wedge \ldots \wedge U_m \rightarrow V_1 \vee \ldots \vee V_n \in \mathcal{C}$, and 2. a mapping σ from variables N_V to the individuals of $\mathcal{A}$ exists such that 2.1 $\sigma(x)$ is not indirectly blocked for each variable $x \in N_V$, 2.2 $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$, and 2.3 $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$, then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$ if $n = 0$; $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ otherwise.
$\geq$-rule	If 1. $\geq n\, R.C(s) \in \mathcal{A}$, 2. s is not blocked in $\mathcal{A}$, and 3. $\mathcal{A}$ does not contain individuals $u_1, \ldots, u_n$ such that 3.1 $\{\mathsf{ar}(R, s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$, and 3.2 either s is blockable or no u_i, $1 \leq i \leq n$, is indirectly blocked in $\mathcal{A}$ 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 distinct successors of s.
$\approx$-rule	If 1. $s \approx t \in \mathcal{A}$ (the equality can possibly be annotated), and 2. $s \neq t$ then $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(s \rightarrow t)$ if t is a named individual, or t is a root individual and s is not a named individual, or s is a descendant of t; $\mathcal{A}_1 := \mathsf{merge}_{\mathcal{A}}(t \rightarrow s)$ otherwise.
$\bot$-rule	If $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\bot\}$.
NI-rule	If 1. $s \approx t\, @^u_{\leq n\, R.B} \in \mathcal{A}$ (the symmetry of $\approx$ applies as usual), 2. u is a root individual, 3. s is a blockable individual that is not a successor of u, and 4. t is a blockable individual then $\mathcal{A}_i := \mathsf{merge}_{\mathcal{A}}(s \rightarrow \lVert u.\langle R, B, i\rangle \rVert_{\mathcal{A}})$ for each $1 \leq i \leq n$.

(4) If $\mathcal{A}$ contains $\geq n\, R.B(s)$, then B is not a nominal guard concept.

(5) If $\mathcal{A}$ contains a blockable individual $s.i$ in some assertion, then $\mathcal{A}$ must contain an assertion of the form $R(s, s.i)$ or $R(s.i, s)$.

(6) $\mathcal{A}$ contains at least one assertion or (in)equality.

Clearly, each initial ABox is an HT-ABox. We now prove that, given an HT-ABox, our calculus produces only HT-ABoxes.

Lemma 12 (HT-Preservation). *For $\mathcal{C}$ a set of HT-clauses and $\mathcal{A}$ an HT-ABox, each ABox $\mathcal{A}'$ obtained by applying a derivation rule to $\mathcal{C}$ and $\mathcal{A}$ is an HT-ABox.*

PROOF. By analyzing each derivation rule from Table 5, we show that $\mathcal{A}'$ satisfies the conditions of HT-ABoxes.

(*Hyp*-rule) Consider an application of the *Hyp*-rule to an HT-clause r of type (14) with a mapping σ, deriving an assertion $\sigma(V)$.

Assume that V is of the form $y_i \approx y_j\, @^x_{\leq k\, R.B}$, so $\sigma(V)$ is of the form $s \approx t\, @^u_{\leq k\, R.B}$. By Definition 7, the antecedent of r then contains atoms of the form $\mathsf{ar}(R, x, y_i)$ and $\mathsf{ar}(R, x, y_j)$ so, by the precondition of the *Hyp*-rule, $\mathcal{A}$ contains assertions $\mathsf{ar}(R, u, s)$

Table 6
Cases in Application of the *Hyp*-Rule to Role Assertions

$\mathsf{ar}(R,u,s)$	$\mathsf{ar}(R,u,t)$	$s \approx t \, @^{u}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,a)$	$\mathsf{ar}(R,v,b)$	$a \approx b \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,a)$	$\mathsf{ar}(R,v,v.n)$	$a \approx v.n \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,a)$	$\mathsf{ar}(R,v,v)$	$a \approx v \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v.n,a)$	$\mathsf{ar}(R,v.n,v)$	$a \approx v \, @^{v.n}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,v.m)$	$\mathsf{ar}(R,v,v.n)$	$v.m \approx v.n \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,v.m)$	$\mathsf{ar}(R,v,v)$	$v.m \approx v \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v.n,v.n.m)$	$\mathsf{ar}(R,v.n,v)$	$v.n.m \approx v \, @^{v.n}_{\leq k\, R.B}$
$\mathsf{ar}(R,v,v)$	$\mathsf{ar}(R,v,v)$	$v \approx v \, @^{v}_{\leq k\, R.B}$
$\mathsf{ar}(R,v.n,v.n)$	$\mathsf{ar}(R,v.n,v)$	$v.n \approx v \, @^{v.n}_{\leq k\, R.B}$
$\mathsf{ar}(R,v.n,v)$	$\mathsf{ar}(R,v.n,v)$	$v \approx v \, @^{v.n}_{\leq k\, R.B}$

and $\mathsf{ar}(R,u,t)$. If u is a root individual and either s or t is a blockable individual that is not a successor of u, then $\sigma(V)$ clearly satisfies Property (2) of HT-ABoxes. Otherwise, since $\mathcal{A}$ satisfies Property (1) of HT-ABoxes, we have these possibilities shown in Table 6, for v a blockable individual, and a and b root individuals. For brevity, we omit the symmetric combinations where the roles of $\mathsf{ar}(R,u,s)$ and $\mathsf{ar}(R,u,t)$ are exchanged. Clearly, $\sigma(V)$ satisfies Property (2) of HT-ABoxes.

Assume that V is of the form $x \approx z_j$, so $\sigma(V)$ is of the form $s \approx t$. By Definition 7, the antecedent of r then contains an atom $O_a(z_j)$, so either $O_a(s) \in \mathcal{A}$ or $O_a(t) \in \mathcal{A}$. By Property (3) of HT-ABoxes, either s or t is a named individual, so $\sigma(V)$ satisfies Property (2) of HT-ABoxes.

Assume that V is of the form $R(x,x)$. Then, $\sigma(V)$ is of the form $R(s,s)$, and it satisfies Property (1) of HT-ABoxes.

Assume that V is of the form $R(x,y_i)$ or $R(y_i,x)$, so $\sigma(V)$ is of the form $R(s,t)$. By Definition 7, the antecedent of r then contains an atom of the form $S(x,y_i)$ or $S(y_i,x)$, and either $S(s,t) \in \mathcal{A}$ or $S(t,s) \in \mathcal{A}$; these assertions satisfy Property (1) of HT-ABoxes, so $R(s,t)$ satisfies it as well.

Assume that V is of the form $R(x,z_j)$ or $R(z_j,x)$, so $\sigma(V)$ is of the form $R(s,t)$. By Definition 7, the antecedent of r then contains an atom of the form $O_a(z_j)$ for O_a a nominal guard concept, and either $O_a(s) \in \mathcal{A}$ or $O_a(t) \in \mathcal{A}$; by Property (3) of HT-ABoxes, either s or t is a named individual, so $R(s,t)$ satisfies Property (1) of HT-ABoxes.

Assume that V is of the form $C(x)$ of $C(y_i)$, so $\sigma(V)$ is of the form $C(s)$. By Definition 7, C is for the form B or $\geq n\, S.B$, for B a literal but not a nominal guard concept. Clearly, $\sigma(V)$ satisfies Property (3) of HT-ABoxes.

($\geq$-rule) Consider an application of the $\geq$-rule to an assertion $\geq n\, R.C(s)$. By Property (4) of HT-ABoxes, C is not a nominal guard concept, so all assertions $C(t_i)$ introduced by the rule satisfy Property (3) of HT-ABoxes. Furthermore, all t_i introduced by the rule are fresh blockable successors of s, and all role assertions introduced by the rule are of the form $R(s,t_i)$ or $R(t_i,s)$, so they satisfy Properties

Table 7
Cases in Application of the $\approx$-Rule to Role Assertions

$R(s,u)$	$s \approx t$	$R(t,u)$
$R(v.i,v)$	$v.i \approx v.j$	$R(v.j,v)$
$R(v.i,v)$	$v.i \approx v$	$R(v,v)$
$R(t.j.i,t.j)$	$t.j.i \approx t$	$R(t,t.j)$
$R(v.i,v.i)$	$v.i \approx v.j$	$R(v.j,v.j)$
$R(v.i,v.i)$	$v.i \approx v$	$R(v,v)$
$R(t.j.i,t.j.i)$	$t.j.i \approx t$	$R(t,t)$

(1) and (5) of HT-ABoxes. The inequalities introduced by the rule satisfy the properties of HT-ABoxes vacuously.

($\approx$-rule) Consider an application of the $\approx$-rule to a possibly annotated equality $s \approx t$, where s is merged into t (the annotation of this equality plays no role in the rule application). By the conditions on the $\mapsto$ relation of $\mathcal{A}$, the ABox $\mathcal{A}$ contains no renaming for s or t, so the renaming $s \mapsto t$ is the only renaming for s in $\mathcal{A}'$, and adding this renaming to $\mathcal{A}$ does not introduce a cycle in $\mapsto$. Merging replaces all occurrences of s in $\mathcal{A}$, so no assertion or (in)equality of $\mathcal{A}'$ contains s. Hence, $\mathcal{A}'$ satisfies the conditions on the $\mapsto$ relation.

The *NI*-rule is not applicable to $s \approx t$ by the rule precedence, so $s \approx t$ can be of the form $v \approx a$, $v.i \approx v.j$, $v.i \approx v$, or $v.i.j \approx v$ for $a \in N_O$ and $v \in N_A$; we denote this property with (*). Pruning removes all successors of s, so $\mathcal{A}'$ satisfies Property (5) of HT-ABoxes. We next consider the types of assertions of $\mathcal{A}$ that change when s is merged into t.

Consider a role assertion $R(s,u) \in \mathcal{A}$ that is changed into $R(t,u) \in \mathcal{A}'$. If either t or u is a root individual, then $R(t,u)$ clearly satisfies Property (1) of HT-ABoxes, so assume that t and u are both blockable individuals. Then, u is not a successor of s, since the $\approx$-rule prunes all assertions that contain a descendant of the merged individual. But then, since $R(s,u)$ satisfies Property (1) of HT-ABoxes and by (*), we have the possibilities shown in Table 7. The cases when $R(u,s) \in \mathcal{A}$ is changed by merging into $R(u,t) \in \mathcal{A}'$ are analogous, so we can conclude that $R(u,t)$ satisfies Property (1) of HT-ABoxes.

For a a root individual, $s \approx u \,@^{a}_{\leq n\, R.C}$ can be changed into $t \approx u \,@^{a}_{\leq n\, R.C}$, and $s \approx a$ can be changed into $t \approx a$. In both cases, the resulting equality satisfies Property (2) of HT-ABoxes. For the remaining cases, assume that a possibly annotated equality $s \approx u$ is changed into a possibly annotated equality $t \approx u$. If s is a root individual, then t and u are root individuals, so $t \approx u$ satisfies Property (2) of HT-ABoxes. Assume that s is a blockable individual. Since the $\approx$-rule prunes all assertions that contain a descendant of the merged individual, u is not a successor of s. By (*), Property (2) of HT-ABoxes, and the fact that the *NI*-rule is not applicable to $\mathcal{A}$, we have the possibilities shown in Table 8. In all cases, the resulting assertion satisfies Property (2) of HT-ABoxes. Furthermore, replacing s with t in $s \approx t \in \mathcal{A}$ results in $t \approx t \in \mathcal{A}'$, so $\mathcal{A}'$ satisfies Property (6) of HT-ABoxes.

Table 8
Cases in Application of the $\approx$-Rule to Equalities

$s \approx u$	$s \approx t$	$t \approx u$
$v.i \approx v.k$	$v.i \approx v.j$	$v.j \approx v.k$
$v.i \approx v$	$v.i \approx v.j$	$v.j \approx v$
$u.k.i \approx u$	$u.k.i \approx u.k.j$	$u.k.j \approx u$
$v.i \approx v.k$	$v.i \approx v$	$v \approx v.k$
$v.i \approx v$	$v.i \approx v$	$v \approx v$
$u.k.i \approx u$	$u.k.i \approx u.k$	$u.k \approx u$
$t.j.i \approx t.j.k$	$t.j.i \approx t$	$t \approx t.j.k$
$t.j.i \approx t.j$	$t.j.i \approx t$	$t.j \approx t$
$t.j.i \approx t$	$t.j.i \approx t$	$t \approx t$

Consider an assertion $C(s) \in \mathcal{A}$ that is changed into $C(t) \in \mathcal{A}'$. If C is a nominal guard concept O_a, then s is a named individual by Property (3) of HT-ABoxes. The $\approx$-rule replaces named individuals only with other named individuals, so t is a named individual as well. Thus, $C(t)$ satisfies Property (3) of HT-ABoxes.

(*NI*-rule) Consider an application of the *NI*-rule to an equality $s \approx t \, @^{u}_{\leq n\, R.B}$ that merges s into a root individual $\|s.\langle R, B, i\rangle\|_{\mathcal{A}}$. The individual s is blockable, so no renaming is added to $\mathcal{A}$ and $\mathcal{A}'$ satisfies the conditions on the $\mapsto$ relation. Since s is replaced by a root individual in role and equality assertions, all resulting assertions satisfy Properties (1) and (2) of HT-ABoxes. Since s is not a named individual, no assertion involving a nominal guard concept is affected by merging, so $\mathcal{A}'$ satisfies Property (3). Pruning removes all successors of s, so $\mathcal{A}'$ satisfies Property (5) of HT-ABoxes. Finally, $\mathcal{A}'$ is clearly not empty, so it satisfies Property (6). $\square$

If $\mathcal{C}$ does not contain atoms of the form $R(x, x)$—that is, if $\mathcal{C}$ has been obtained from a $\mathcal{SHOIQ}$ knowledge base—the proof of Lemma 12 reveals that Definition 11 need not allow for assertions of the forms $R(s, s)$ and $s.i \approx s$. This, in turn, allows us to drop the requirement that $\mathcal{L}_{\mathcal{A}}(s, s) = \mathcal{L}_{\mathcal{A}}(t, t)$ in the pair-wise blocking condition from Definition 10: if s and t are blockable individuals, then $\mathcal{L}_{\mathcal{A}'}(s, s) = \mathcal{L}_{\mathcal{A}'}(t, t) = \emptyset$ for any ABox $\mathcal{A}'$ obtained from $\mathcal{A}$ and $\mathcal{C}$. Thus, the blocking condition in such cases is the same as in [21,23,30].

We next prove soundness and completeness of our calculus. We use these notions as it is customary in resolution-based theorem proving: a calculus is sound if it is possible to apply its inference such that the satisfiability of a theory is preserved; furthermore, it is complete if, whenever the calculus terminates without detecting a contradiction, the theory is indeed satisfiable.

Lemma 13 (Soundness). *Let $\mathcal{C}$ be a set of HT-clauses and $\mathcal{A}$ an input ABox such that $(\mathcal{C}, \mathcal{A})$ is satisfiable. Then, each derivation for $\mathcal{C}$ and $\mathcal{A}$ contains a leaf node t such that $\lambda(t)$ is clash-free.*

PROOF. We say that a model I of an ABox $\mathcal{A}_0$ is *NI-compatible* with $\mathcal{A}_0$ if the following conditions are satisfied:

- For each root individual a occurring in $\mathcal{A}_0$, we have
  - $a^I = (\|a\|_{\mathcal{A}_0})^I$, and
  - $a^I \in (\leq n\, R.B)^I$, $\langle a^I, \alpha\rangle \in R^I$, and $\alpha \in B^I$ for some $\alpha \in \triangle^I$, some role R, and some literal concept B imply $\alpha = (a.\langle R, B, i\rangle)^I$ for some $1 \leq i \leq n$.
- If $s \approx t \,@^u_{\leq n\, R.B} \in \mathcal{A}_0$, then we have $\langle u^I, s^I\rangle \in R^I$, $\langle u^I, t^I\rangle \in R^I$, $s^I \in B^I$, $t^I \in B^I$, and $u^I \in (\leq n\, R.B)^I$.

To prove this lemma, we first show the following property (*): if $(\mathcal{C}, \mathcal{A}_0)$ is satisfiable in a model that is *NI*-compatible with $\mathcal{A}_0$ and $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are ABoxes obtained by applying a derivation rule to $\mathcal{C}$ and $\mathcal{A}_0$, then some $(\mathcal{C}, \mathcal{A}_i)$ is satisfiable in a model that is *NI*-compatible with $\mathcal{A}_i$. Let I be a model of $(\mathcal{C}, \mathcal{A}_0)$ that is *NI*-compatible with $\mathcal{A}_0$ and consider all possible derivation rules that can derive $\mathcal{A}_1, \ldots, \mathcal{A}_n$ from $\mathcal{A}_0$ and $\mathcal{C}$.

(*Hyp*-rule) Consider an application of the *Hyp*-rule to an HT-clause of the form (14). Since $\sigma(U_i) \in \mathcal{A}_0$, 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 $\mathcal{A}_j := \mathcal{A}_0 \cup \{\sigma(V_j)\}$, we have $I \models (\mathcal{C}, \mathcal{A}_j)$.

If $I \models \sigma(V_j)$ for some atom V_j not of the form $\psi = y_k \approx y_\ell \,@^x_{\leq h\, R.B}$, then I is clearly *NI*-compatible with $\mathcal{A}_j$. Furthermore, for each V_j of the form ψ, clearly $\langle \sigma(x)^I, \sigma(y_k)^I\rangle \in R^I$, $\langle \sigma(x)^I, \sigma(y_\ell)^I\rangle \in R^I$, $\sigma(y_k)^I \in B^I$, and $\sigma(y_\ell)^I \in B^I$. We denote these two properties with (**).

Assume that I is not *NI*-compatible with $\mathcal{A}_j$ for each $1 \leq j \leq n$. By (**), then $I \not\models \sigma(V_j)$ for each V_j not of the form ψ, and $\sigma(x)^I \notin (\leq h\, R.B)^I$ for each V_j of the form ψ. Let $\mu : N_V \to \triangle^I$ be a variable mapping such that $\mu(x) = \sigma(x)^I$ and $\mu(y_k) = \sigma(y_k)^I$ for each branch variable y_k not occurring in an atom of the form ψ; furthermore, for each set of branch variables $y_1, \ldots, y_{h+1}$ occurring in an atom of the form ψ, we set $\mu(y_1), \ldots, \mu(y_{h+1})$ to arbitrarily chosen domain elements that verify $\sigma(x)^I \notin (\leq h\, R.B)^I$. Clearly, $I, \mu \not\models V_j$ for each V_j not occurring in a subset (15) or (16) of r; furthermore, by the definition of μ, we have that $I, \mu \not\models V_j$ for each V_j occurring in a subset of (15) or (16) of r. But then, we conclude $I, \mu \not\models (\mathcal{C}, \mathcal{A}_0)$, which is a contradiction.

($\geq$-rule) Since $\geq n\, R.C(s) \in \mathcal{A}_0$, we have $I \models\, \geq n\, R.C(s)$, so domain elements $\alpha_1, \ldots, \alpha_n \in \triangle^I$ exist where $\langle s^I, \alpha_i\rangle \in R^I$ 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 an interpretation 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 $I' \models (\mathcal{C}, \mathcal{A}_1)$. The individuals t_i are not root individuals, so I' is *NI*-compatible with $\mathcal{A}_1$.

($\approx$-rule) Assume that the $\approx$-rule is applied to the assertion $s \approx t \in \mathcal{A}_0$ and s is merged into t. Since $I \models s \approx t$, we have $s^I = t^I$. Pruning removes assertions, so I is a model of the pruned ABox by monotonicity. Merging simply replaces an individual with a synonym, so $I \models (\mathcal{C}, \mathcal{A}_1)$. Furthermore, $\|s\|_{\mathcal{A}_1} = t$, so I is *NI*-compatible with $\mathcal{A}_1$.

($\bot$-rule) This rule is never applicable if $(\mathcal{C}, \mathcal{A}_0)$ is satisfiable.

(*NI*-rule) Assume that the *NI*-rule is applied to some $s \approx t \,@^u_{\leq n\, R.B} \in \mathcal{A}_0$ and s is merged into a root individual. Since I is *NI*-compatible with $\mathcal{A}_0$, we have

$u^I \in (\leq n\, R.B)^I$, $\langle u^I, s^I \rangle \in R^I$, $s^I \in B^I$, and $s^I = (u.\langle R, B, i \rangle)^I$ for some $1 \leq i \leq n$. Let $v_i = \|u.\langle R, B, i\rangle\|_{\mathcal{A}_0}$; since I is *NI*-compatible, we have $(u.\langle R, B, i \rangle)^I = v_i^I$. Thus, the *NI*-rule replaces s by its synonym v_i, so $I \models (\mathcal{C}, \mathcal{A}_i)$ just like in the case of the $\approx$-rule. If v_i does not occur in $\mathcal{A}_0$, the interpretation I may not be *NI*-compatible with $\mathcal{A}_i$ because it does not interpret $v_i.\langle S, C, \ell \rangle$ correctly. We then extend I to I' as follows. For each m, S, and C such that $v_i \in (\leq m\, S.C)^I$, let $\alpha_1, \ldots, \alpha_k$ be the elements of $\triangle^I$ such that $\langle v_i^I, \alpha_j \rangle \in S^I$ and $\alpha_j \in C^I$; clearly, $k \leq m$. We then set $(v_i.\langle S, C, \ell \rangle)^{I'} = \alpha_\ell$ for $1 \leq \ell \leq k$. Since none of $v_i.\langle S, C, \ell\rangle$ occurs in $\mathcal{A}_i$, we have $I' \models (\mathcal{C}, \mathcal{A}_i)$, so I' is *NI*-compatible with $\mathcal{A}_j$.

This completes the proof of (*). To prove the main claim of this lemma, let $\mathcal{A}$ be an input ABox. Similarly as for the *NI*-rule in the proof of (*), we can extend I to a model I' of $(\mathcal{C}, \mathcal{A})$. Since $\mathcal{A}$ does not contain at-most equalities, I' is *NI*-compatible with $\mathcal{A}$. The claim of this lemma then follows by a straightforward inductive application of (*). $\square$

Lemma 14 (Completeness)**.** *If a derivation for a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$ exists in which some leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{C}, \mathcal{A})$ is satisfiable.*

PROOF. We prove the lemma by unraveling $\mathcal{A}'$ into a model of $(\mathcal{C}, \mathcal{A})$. To enable this, we first define several useful notions.

A *path* is a finite sequences of pairs of individuals of the form $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. Let $\mathsf{tail}(p) = x_n$, $\mathsf{tail}'(p) = x'_n$, and let $q = [p \mid \frac{x_{n+1}}{x'_{n+1}}]$ 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:

- $[\frac{a}{a}] \in \mathcal{P}(\mathcal{A}')$ for each root individual a occurring in $\mathcal{A}'$;
- $[p \mid \frac{s'}{s'}] \in \mathcal{P}(\mathcal{A}')$ if $p \in \mathcal{P}(\mathcal{A}')$, s' is a successor of $\mathsf{tail}(p)$, s' occurs in $\mathcal{A}'$, and s' is not blocked in $\mathcal{A}'$; and
- $[p \mid \frac{s}{s'}] \in \mathcal{P}(\mathcal{A}')$ if $p \in \mathcal{P}(\mathcal{A}')$, s' is a successor of $\mathsf{tail}(p)$, s' occurs in $\mathcal{A}'$, and s' is directly blocked in $\mathcal{A}'$ by s.

By the definition of blocking, the following three properties (*) hold for each path $p \in \mathcal{P}(\mathcal{A}')$: $C(\mathsf{tail}(p)) \in \mathcal{A}'$ if and only if $C(\mathsf{tail}'(p)) \in \mathcal{A}'$ for a blocking-relevant concept C; $R(\mathsf{tail}(p), \mathsf{tail}(p)) \in \mathcal{A}'$ if and only $R(\mathsf{tail}'(p), \mathsf{tail}'(p)) \in \mathcal{A}'$ for an atomic role R; and $\mathsf{tail}(p)$ is not blocked.

Let I be the interpretation constructed as follows.

$$\begin{aligned}
\triangle^I &= \mathcal{P}(\mathcal{A}') \\
a^I &= [\tfrac{a}{a}] \text{ for each root individual } a \text{ that occurs in } \mathcal{A}' \\
a^I &= b^I \text{ if } a \neq b \text{ and } \|a\|_{\mathcal{A}'} = b \\
A^I &= \{p \mid A(\mathsf{tail}(p)) \in \mathcal{A}'\} \\
R^I &= \{\langle [\tfrac{a}{a}], p\rangle \mid a \text{ is a root individual and } R(a, \mathsf{tail}(p)) \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle p, [\tfrac{a}{a}]\rangle \mid a \text{ is a root individual and } R(\mathsf{tail}(p), a) \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle p, [p \mid \tfrac{s}{s'}]\rangle \mid R(\mathsf{tail}(p), s') \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle [p \mid \tfrac{s}{s'}], p\rangle \mid R(s', \mathsf{tail}(p)) \in \mathcal{A}'\} \cup \\
&\quad\ \{\langle p, p\rangle \mid R(\mathsf{tail}(p), \mathsf{tail}(p)) \in \mathcal{A}'\}
\end{aligned}$$

$\mathcal{A}'$ is an HT-ABox, 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 $s' = t'$:* Immediate, since the $\approx$-rule is not applicable to $\mathcal{A}'$. This property holds even if $s' \approx t'$ is annotated.
- *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 $p_s \neq p_t$.
- *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 $1 \leq i < j \leq n$. Each assertion $\mathsf{ar}(R, s, u_i)$ satisfies Property (1) of HT-ABoxes, so each u_i can be of one of the following forms.
  - $u_i = s$. In this case, let $p_{u_i} = p_s$.
  - u_i is a successor of s. If u_i is directly blocked by v_i, let $p_{u_i} = [p_s \mid \frac{v_i}{u_i}]$; 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, we have $C(\mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$ and $\mathsf{ar}(R, s', \mathsf{tail}'(p_{u_i})) \in \mathcal{A}'$.
  - u_i and s do not satisfy any of the previous three conditions. If s is a blockable individual, then u_i is a root individual, so let $p_{u_i} = [\frac{u_i}{u_i}]$. If s is a root individual, then u_i is not indirectly blocked in $\mathcal{A}'$ by Condition 3.2 of the $\geq$-rule; but then, none of the ancestors of u_i are blocked in $\mathcal{A}'$, so we can choose some $p_{u_i} \in \triangle^I$ of the form $p_{u_i} = [p \mid \frac{u_i}{u_i}]$ for some path p.

  In all these 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 $p_{u_i} \neq p_{u_j}$ holds. Furthermore, 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 $\mathcal{A}'$ contains renamings that, when applied to α, produce an assertion $\alpha' \in \mathcal{A}'$. But then $I \models \alpha$ by the definition of I.

It remains to be shown that $I \models \mathcal{C}$. Consider each HT-clause $r \in \mathcal{C}$ containing atoms of the form $A_i(x)$, $U_k(x,x)$, $\mathsf{ar}(R_i,x,y_i)$, $B_i(y_i)$, and $C_j(z_j)$ in the antecedent. Furthermore, consider a variable mapping μ such that the antecedent of r is true in I and μ—that is, $p_x \in A_i^I$, $\langle p_x,p_x\rangle \in U_k^I$, $\langle p_x,p_{y_i}\rangle \in R_i^I$, $p_{y_i} \in B_i^I$, and $p_{z_j} \in C_j^I$ for $p_x = \mu(x)$, $p_{y_i} = \mu(y_i)$, and $p_{z_j} = \mu(z_j)$. Let $s = \mathsf{tail}(p_x)$, $s' = \mathsf{tail}'(p_x)$, $t_i' = \mathsf{tail}'(p_{y_i})$. By the definition of I, we have $A_i(s) \in \mathcal{A}'$ and $U_k(s,s) \in \mathcal{A}'$. We define t_i as follows.

- If t_i' is a blocked successor of s, let $t_i = t_i'$. By the definition of I, we have $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i,s,t_i) \in \mathcal{A}'$.
- If s' is blocked and t_i' is the predecessor of s', let t_i be the predecessor of s. Such t_i exists since s blocks s' and therefore s is blockable. By the definition of I, we have $B_i(t_i') \in \mathcal{A}'$ and $\mathsf{ar}(R_i,s',t_i') \in \mathcal{A}'$; furthermore, by the definition of blocking, we have $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i,s,t_i) \in \mathcal{A}'$.
- Otherwise, let $t_i = \mathsf{tail}(p_{y_i})$. By the definition of I, we have $B_i(t_i) \in \mathcal{A}'$ and $\mathsf{ar}(R_i,s,t_i) \in \mathcal{A}'$.

By Definition 7, the antecedent of r contains an atom of the form $O_a(z_j)$ for each nominal variable z_j. Thus, by the definition of I and Property (3) of HT-Bboxes, we have p_{z_j} is of the form $[\frac{u_j}{u_j}]$ for u_j a named individual; furthermore, $C_j(u_j) \in \mathcal{A}'$.

Let σ be a mapping such that $\sigma(x) = s$, $\sigma(y_i) = t_i$, and $\sigma(z_j) = u_j$. Clearly, neither s nor t_i are indirectly blocked. The *Hyp*-rule is not applicable to r, $\mathcal{A}'$, and σ, so r contains an atom V_i in the consequent such that $\sigma(V_i) \in \mathcal{A}'$. Depending on the type of V_i, we have the following possibilities.

Assume that V_i is of the form $y_i \approx y_j \ @^x_{\leq k\, S.B}$, so $t_i \approx t_j \ @^s_{\leq k\, S.B} \in \mathcal{A}'$. By (**), we have $t_i = t_j$. By Definition 7, r contains a subclause of the form (15) or (16), so the antecedent of r contains atoms $\mathsf{ar}(S,x,y_i)$ and $\mathsf{ar}(S,x,y_j)$; therefore, we have $\langle p_x,p_{y_i}\rangle \in S^I$ and $\langle p_x,p_{y_j}\rangle \in S^I$. The *NI*-rule is not applicable to $t_i \approx t_j \ @^s_{\leq k\, S.B}$ so, if s is a root individual, then t_i is either a root individual or a successor of s. Thus, by Property (1) of HT-ABoxes, we have these possibilities: t_i is a root individual, $t_i = s$, t_i is a predecessor of s, or t_i is a successor of s. If t_i is a root individual or if $t_i = s$, then $t_i = t_j$ clearly implies $p_{y_i} = p_{y_j}$. The paths p_{y_i} and p_{y_j} can be both successors of p_x, but again, $t_i = t_j$ implies $p_{y_i} = p_{y_j}$. The paths p_{y_i} and p_{y_j} can be both predecessors of p_x; but then $p_{y_i} = p_{y_j}$ since p_x can have at most one predecessor. Assume that p_{y_i} is a predecessor, but p_{y_j} is a successor of p_x; since t_i is not blocked, it must be that $t_i \neq t_j$, which is a contradiction. Thus, $I, \mu \models r$.

Assume that V_i is an equality of the form $x \approx z_j$, so $s \approx u_j \in \mathcal{A}$. By (**), we have $s = u_j$; since u_j is a named individual, this implies $p_x = p_{z_j}$, so $I, \mu \models r$.

Assume that V_i is of the form $D_i(x)$, $E_i(y_i)$, or $R_i(x,x)$, so we have $D_i(s) \in \mathcal{A}'$, $E_i(t_i) \in \mathcal{A}'$, or $R_i(s,s) \in \mathcal{A}'$. By the definition of blocking, we have $D_i(s') \in \mathcal{A}'$, $E_i(t'_i) \in \mathcal{A}'$, or $R_i(s',s') \in \mathcal{A}'$. Finally, by (**) and the definition of I, we have $p_s \in D_i^I$, $p_{y_i} \in E_i^I$, or $\langle p_x, p_x \rangle \in R_i^I$, so $I, \mu \models r$.

Assume that V_i is of the form $\mathsf{ar}(S_i, x, y_i)$, so $\mathsf{ar}(S_i, s, t_i) \in \mathcal{A}'$. By the definition of blocking, we have $\mathsf{ar}(S_i, s', t'_i) \in \mathcal{A}'$. Finally, by the definition of I, we have $\langle p_x, p_{y_i} \rangle \in S_i^I$, so $I, \mu \models r$.

Assume that V_i is of the form $\mathsf{ar}(S_j, x, z_j)$, so $\mathsf{ar}(S_j, s, u_j) \in \mathcal{A}'$. Since u_j is a named individual, by the definition of I we have $\langle p_x, p_{z_j} \rangle \in S_j^I$, so $I, \mu \models r$. $\square$

We next prove termination of our calculus.

Lemma 15 (Termination). *For a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$, let $|\mathcal{C}, \mathcal{A}|$ be the sum of the size of $\mathcal{A}$, the number of concepts and roles in $\mathcal{C}$, and the numbers occurring in $\mathcal{C}$ in atoms of the form $\geq n\, R.B$ and $y_i \approx y_j\ @^x_{\leq n\, R.B}$. Then, the total number of individuals introduced on each derivation path is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$; furthermore, each derivation is finite.*

PROOF. We prove the claim by showing that *(i)* each individual can participate in at most exponentially many rule applications, and *(ii)* the number of new individuals introduced on each path of a derivation is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$. The supply of blockable individuals is infinite, so we can assume that no blockable individual is introduced twice on a derivation path. Furthermore, if the root individual s is removed from an ABox $\mathcal{A}'$ due to merging, then a renaming is added to $\mathcal{A}'$ that ensures that $\|s\|_{\mathcal{A}'} \neq s$. Once a renaming is added to $\mathcal{A}'$, all ABoxes occurring below $\mathcal{A}'$ in a derivation will contain this renaming as well, so no subsequent application of the *NI*-rule can reintroduce s.

Next, we prove *(i)*—that is, that each derivation rule can be applied at most an exponential number of times to a fixed set of individuals in a derivation path—by considering each derivation rule.

- An application of the *Hyp*-rule to an HT-clause r of the form (14) and a mapping σ introduces an assertion $\sigma(V_i)$, which prevents a subsequent reapplication of the *Hyp*-rule to the same r and σ. Merging and pruning can remove $\sigma(V_i)$ in subsequent derivation steps, but such an inference also removes at least one individual occurring in σ, preventing the reuse of the same σ in a future application of the *Hyp*-rule.
- An application of the $\geq$-rule to an assertion $\geq n\, R.C(s)$ introduces $t_1, \ldots, t_n$ as fresh successors of s and the assertions $C(t_i)$, $\mathsf{ar}(R, s, t_i)$, and $t_i \not\approx t_j$ for $1 \leq i < j \leq n$. Thus, the individuals $u_1, \ldots, u_n$ from the Condition 3 of the $\geq$-rule can be matched to $t_1, \ldots, t_n$. Furthermore, if s is not blockable, then it is a root individual, so none of t_i can become indirectly blocked and Condition 3.2 is always satisfied for t_i. If some t_i is merged into another individual v, then $C(v)$, $\mathsf{ar}(R, s, v)$, and $v \not\approx t_j$ are added to the ABox, so the ABox still contains individuals that can be matched to Condition 3 of the $\geq$-rule.

- An application of the $\approx$-rule to $s \approx t$ removes either s or t, so the rule cannot be reapplied to the same s and t
- An application of the $\perp$-rule produces an ABox that labels a derivation leaf.
- An application of the *NI*-rule to an equality $s \approx t \mathbin{@}^{u}_{\leq n\, R.B}$ removes s, so the rule cannot be reapplied to the same $\leq n\, R.B$, s and u.

Next, we prove *(ii)*—that is, that the total number of individuals introduced on a derivation path is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$. A *path of length n* between individuals s and t in an ABox $\mathcal{A}'$ is a sequence of individuals $u_0, u_1, \ldots, u_n$ such that $u_0 = s$, $u_n = t$, and, for each $0 \leq i \leq n-1$, either $R(u_i, u_{i+1}) \in \mathcal{A}'$ or $R(u_{i+1}, u_i) \in \mathcal{A}'$ for some atomic role R.

A *root path* for a root individual t in an ABox $\mathcal{A}'$ is a path between t and a named individual s such that all intermediate individuals u_i, $1 \leq i \leq n-1$, are root individuals. The *level* $\mathsf{lev}(t)$ of t is the length of the shortest root path for t.

The *depth* $\mathsf{dep}(t)$ of an individual t is the number of the ancestors of t; thus, the depth of each root individual is 0. Due to Property (5) of HT-ABoxes, if an individual t occurs in an ABox $\mathcal{A}'$, then $\mathcal{A}'$ contains a path of length $\mathsf{dep}(t)$ between a root individual s and t such that all intermediate individuals u_i, $1 \leq i \leq n-1$, are all the ancestors of t.

We now show that the maximum level of a root individual and the maximum depth of every individual are both at most exponential in the size of $\mathcal{C}$ and $\mathcal{A}$.

We first note that an application of an inference rule never increases the level of an individual. This is because a named individual is never pruned and can be merged only into other named individuals,[6] and a root individual can be merged only into another root individual. Such rule applications can make a root path only shorter, and not longer.

Let m be the number of concepts and n the number of atomic roles that occur in $\mathcal{A}$ and $\mathcal{C}$, let $\gamma = 2^{2m+3n} + 1$, and let $\mathcal{A}'$ be an ABox labeling a node of a derivation for $\mathcal{A}$ and $\mathcal{C}$. We next show that (1) $\mathsf{dep}(t) \leq \gamma$ for each individual t occurring in $\mathcal{A}'$, and (2) if t is a root individual, then $\mathsf{lev}(t) \leq \gamma$.

(Claim 1) For a pair of individuals s and t occurring in $\mathcal{A}'$, there are 2^m different labels $\mathcal{L}_{\mathcal{A}'}(s)$ and 2^n different labels $\mathcal{L}_{\mathcal{A}'}(s,t)$. Thus, if $\mathcal{A}'$ contains at least $\gamma = 2^m \cdot 2^m \cdot 2^n \cdot 2^n \cdot 2^n + 1$ predecessor-successor pairs of blockable individuals, then $\mathcal{A}'$ must contain two pairs $\langle s, s.i\rangle$ and $\langle t, t.j\rangle$ such that the following conditions are satisfied:

$$\begin{array}{ccc} \mathcal{L}_{\mathcal{A}'}(s.i) = \mathcal{L}_{\mathcal{A}'}(t.j) & \mathcal{L}_{\mathcal{A}'}(s) = \mathcal{L}_{\mathcal{A}'}(t) & \mathcal{L}_{\mathcal{A}'}(s.i, s.i) = \mathcal{L}_{\mathcal{A}'}(t.j, t.j) \\ \mathcal{L}_{\mathcal{A}'}(s, s.i) = \mathcal{L}_{\mathcal{A}'}(t, t.j) & \mathcal{L}_{\mathcal{A}'}(s.i, s) = \mathcal{L}_{\mathcal{A}'}(t.j, t) & \end{array}$$

Since $\prec$ contains the descendant relation, a path in $\mathcal{A}'$ containing γ blockable individuals must include at least one blocked individual, so a blockable individual

[6] If a derivation rule replaced a named individual with an individual that is not named, the levels of other root individuals could increase.

of depth γ must be blocked. The $\geq$-rule is applied only to individuals that are not blocked, so the rule cannot introduce an individual u such that $\mathsf{dep}(u) > \gamma$.

(Claim 2) We say that an individual s is a *neighbor* of an individual t in an ABox $\mathcal{A}'$ if s and t occur in $\mathcal{A}'$ in an assertion of the form $R(s,t)$, $R(t,s)$, $s \approx t$, or $t \approx u \,@^{s}_{\leq n\, R.B}$, $s \approx u \,@^{t}_{\leq n\, R.B}$ (the symmetry of $\approx$ applies as usual). We now prove the stronger claim (*): if a root individual s has a blockable neighbor t in $\mathcal{A}'$ that is not a successor of s, then $\mathsf{lev}(s) + \mathsf{dep}(t) \leq \gamma$, and, if s has no blockable neighbors, then $\mathsf{lev}(s) \leq \gamma$. This is clearly true for the ABox $\mathcal{A}$ labeling the root of a derivation, which contains only named individuals. We now assume that (*) holds for some ABox $\mathcal{A}'$ and consider all possible derivation rules that can be applied to $\mathcal{A}'$.

- Assume that the *Hyp*-rule derives an assertion $R(s,t)$ or $R(t,s)$, where s is a root individual and t is a blockable neighbor but not a successor of s. Let $R(x,y)$ or $R(y,x)$ be the atom from the consequent of an HT-clause r that is instantiated by the inference rule. Then, the antecedent of r contains either an atom of the form $S(x,y)$ or $S(y,x)$ that is matched to an assertion of the form $S(s,t)$ or $S(t,s)$ in $\mathcal{A}'$, or an atom of the form $O_a(x)$ or $O_a(y)$ that is matched to an assertion of the form $O_a(s)$. In the first case, since $\mathcal{A}'$ satisfies (*), the resulting ABox satisfies (*) as well. In the second case, $\mathsf{dep}(t) \leq \gamma$ and $\mathsf{lev}(s) = 0$, so the resulting ABox satisfies (*) as well.
- Assume that the *Hyp*-rule derives an assertion $t \approx u \,@^{s}_{\leq n\, R.C}$ or $s \approx u \,@^{t}_{\leq n\, R.C}$, where s is a root individual and t is a blockable neighbor but not a successor of s. By Definition 7, the antecedent of the HT-clause then contain an atom of the form $\mathsf{ar}(R, x, y_i)$ that is matched to an assertion $\mathsf{ar}(R, s, t)$ or $\mathsf{ar}(R, t, s)$. But then, since $\mathcal{A}'$ satisfies (*), the resulting ABox satisfies (*) as well.
- If the *Hyp*-rule derives an assertion $s \approx t$ where s is a root individual and t is a blockable neighbor but not a successor of s, the only remaining possibility is that the consequent of the HT-clause then contains the equality $x \approx z_j$ or $y_i \approx z_j$. By Definition 7, the antecedent then contains $O_a(z_j)$ that is matched to an assertion $O_a(s)$ where s is a named individual. Since $\mathcal{A}'$ satisfies (*), $\mathsf{dep}(t) \leq \gamma$ and $\mathsf{lev}(s) = 0$, so the resulting ABox satisfies (*) as well.
- Assume that $\geq$-rule introduces a new blockable neighbor t of a root individual s. Then, t is a successor of s, and so the resulting ABox satisfies (*). In all other cases, the rule introduces a successor of a blockable individual, so the resulting ABox satisfies (*) trivially.
- We consider the types of equalities to with the $\approx$-rule can be applied in $\mathcal{A}'$.
  - $u.i \approx u.j$: Then $\mathsf{dep}(u.i) = \mathsf{dep}(u.j)$, so merging $u.i$ into $u.j$ does not change the depth of a blockable neighbor t of a root individual s.
  - $u.i \approx u$: Then $u.i$ is merged into u, so a root individual s that is a neighbor of $u.i$ can become a neighbor of the individual u, which has lower depth.
  - $u.i.j \approx u$: Then $u.i.j$ is merged into u, so a root individual s that is a neighbor of $u.i.j$ can become a neighbor of the individual u, which has lower depth.
  - $u \approx a$ for a a root individual: Then all successors of u are pruned and u is

merged into b. Since $\mathcal{A}'$ satisfies (*), we have $\mathsf{lev}(u) + \mathsf{dep}(a) \leq \gamma$. Hence, a can acquire a blockable predecessor v of u as neighbor. Since $\mathsf{dep}(v) = \mathsf{dep}(u) - 1$, we have $\mathsf{lev}(a) + \mathsf{dep}(v) \leq \gamma$.

  · $a \approx b$ for a and b a root individuals: If a is merged into b, the level of b after merging is the minimum level of a and b before merging. Hence, (*) holds in the resulting ABox for b and each blockable neighbor t inherited from a.

- An application of the $\perp$-rule trivially preserves (*).
- Assume that the *NI*-rule is applied to an assertion $s \approx t \,@^{u}_{\leq n\,R.B}$ such that $s = w.i$ and $u \neq w$. Then, the individual $w.i$ is merged into a root individual $v = \|u.\langle R, B, i\rangle\|_{\mathcal{A}'}$. Thus, the root individual v acquires u as a neighbor, and $\mathsf{lev}(v) \leq \mathsf{lev}(u) + 1$. Furthermore, v acquires w as a neighbor; since $w.i$ is a neighbor but not a successor of u, we have $\mathsf{lev}(u) + \mathsf{dep}(w.i) \leq \gamma$ by the assumption. Finally, since $\mathsf{dep}(w) = \mathsf{dep}(w.i) - 1$, we have $\mathsf{lev}(v) + \mathsf{dep}(w) \leq \gamma$.

We now complete the proof of claim *(ii)*—that is, that the total number of individuals introduced by derivation rules is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

All named individuals are of level 0 and are never introduced by the derivation rules. An application of the *NI*-rule to a root individual u of level ℓ can introduce at most n root individuals of level $\ell + 1$ for each concept $\leq n\,R.B$ that occurs in $\mathcal{C}$. Thus, for each named individual, the derivation rules can create a tree of root individuals. The maximum depth of the tree is γ, which is exponential in $\mathcal{A}$ and $\mathcal{C}$. Furthermore, the maximum branching factor b is equal to the sum of all numbers occurring in $\mathcal{C}$ in atoms of the form $y_i \approx y_j \,@^{x}_{\leq n\,R.B}$. If numbers are coded in binary, then b is exponential in $|\mathcal{C}, \mathcal{A}|$, so each such tree is doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

Similarly, each root individual can become root of a tree of blockable individuals of depth γ. Each blockable individual is introduced by applying the $\geq$-rule to its predecessor. Furthermore, the $\geq$-rule can be applied to an individual s at most once for each concept of the form $\geq n\,R.B$. Thus, the branching factor is exponential assuming binary coding of numbers, and each such tree is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$.

Thus, the total number of individuals appearing in a derivation is at most doubly exponential in $|\mathcal{C}, \mathcal{A}|$. Since the branching factor in the derivation is exponentially bounded by $|\mathcal{C}, \mathcal{A}|$, each derivation is finite. □

We now state the main theorem of this section.

Theorem 16. *Checking whether a $\mathcal{SHOIQ}^+$ knowledge base $\mathcal{K}$ is satisfiable can be performed by computing $\mathcal{K}' = \Delta(\Omega(\mathcal{K}))$ and then checking whether some derivation for $\Xi(\mathcal{K}')$ contains a leaf node labeled with a clash-free ABox. Furthermore, such an algorithm can be implemented in* 2NExpTime *in $|\mathcal{K}|$.*

PROOF. The first part of the theorem follows immediately from Lemmas 3, 6, 13, and 14. By Lemma 15, the total number of individuals is doubly exponential in $|\Xi_{\mathcal{A}}(\mathcal{K}'), \Xi_{\mathcal{TR}}(\mathcal{K}')|$. Since the structural transformation is polynomial, the total

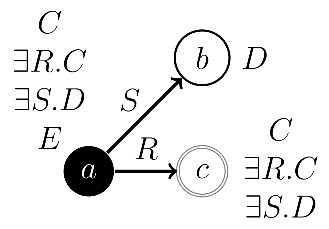


Fig. 11. Problems with Subset Blocking

number of individuals is doubly exponential in $|\mathcal{K}|$. Thus, the existence of a leaf derivation node labeled with a clash-free ABox can be checked by nondeterministically constructing an ABox that is doubly exponential in $|\mathcal{K}|$. □

5 Discussion

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

5.1 *Subset Blocking*

In traditional tableau algorithms for DLs without inverse roles, the pairwise blocking from Definition 10 can improved to simpler *subset blocking* [2].

Definition 17 (Subset Blocking)**.** *Subset blocking is obtained from pairwise blocking (see Definition 10) by changing the notion of direct blocking: a blockable individual s is directly blocked a blockable individual t if and only if t is not blocked, $t \prec s$, and $\mathcal{L}_\mathcal{A}(s) \subseteq \mathcal{L}_\mathcal{A}(t)$.*

Subset blocking is not applicable in the hypertableau setting. Consider the following knowledge base $\mathcal{K}_9$, consisting of an ABox $\mathcal{A}_9$ and a TBox corresponding to the DL-clauses $\mathcal{C}_9$.

$$\begin{aligned} \mathcal{A}_9 &= \{\ C(a)\ \} \\ \mathcal{C}_9 &= \left\{ \begin{array}{ll} C(x) \rightarrow \exists R.C(x), & C(x) \rightarrow \exists S.D(x), \\ S(x,y) \wedge D(y) \rightarrow E(x), & R(x,y) \wedge E(y) \rightarrow \bot \end{array} \right\} \end{aligned} \tag{17}$$

On $\mathcal{K}_9$, the hypertableau algorithm can produce the ABox shown in Figure 11. Individual c is now subset-blocked by a. If, however, we expanded $\exists S.D(c)$ into $S(c,d)$ and $D(d)$, we can derive $E(c)$; together with $R(a,c)$ and the DL-clause $R(x,y) \wedge E(y) \rightarrow \bot$, we get a contradiction.

5.2 *Single Blocking*

For DLs without inverse roles, such as $\mathcal{SHOQ}$, or with inverse roles but without nominals and number restrictions, such as $\mathcal{SHIO}$, the pairwise blocking can be improved to *single blocking*.

Definition 18 (Single Blocking)**.** *Single blocking is obtained from pairwise blocking (see Definition 10) by changing the notion of direct blocking: a blockable individual s is directly blocked a blockable individual t if and only if t is not blocked $t \prec s$, and $\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(t)$.*

The completeness of the hypertableau algorithm with single blocking for $\mathcal{SHOQ}$ can be shown by modifying the model construction in Lemma 14.

Lemma 19. *Let $\mathcal{A}$ be an ABox without inverse roles and $\mathcal{C}$ a set of HT-clauses such that, for each $r \in \mathcal{C}$, (i) r does not contain inverse roles, and (ii) each role atom in r is of the form $R(x, y_i)$ for x the center variable and y_i a branch variable. If a derivation with single blocking for $\mathcal{C}$ and $\mathcal{A}$ exists in which a leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{C}, \mathcal{A})$ is satisfiable.*

PROOF. By slightly modifying the proof of Lemma 12, it is possible to show the following property (*): each atom in $\mathcal{A}'$ involving an atomic role is of the form $R(a, b)$, $R(s, a)$, or $R(s, s.i)$, for a and b named individuals and s a blockable individual. Let $\mathcal{A}''$ be obtained from $\mathcal{A}'$ by removing all assertions containing an indirectly blocked individual. It is easy to see that no derivation rule is applicable to $\mathcal{A}''$ and $\mathcal{C}$. We now construct an interpretation I as follows.

$$\begin{aligned}
\triangle^I &= \{s \mid s \text{ occurs in } \mathcal{A}''\} \\
s^I &= s \text{ for each individual } s \text{ occurring in } \mathcal{A}'' \\
A^I &= \{s \mid A(s) \in \mathcal{A}''\} \\
R^I &= \{\langle s, t\rangle \mid s \text{ is not blocked in } \mathcal{A}'' \text{ and } R(s,t) \in \mathcal{A}''\} \cup \\
&\quad\ \{\langle s, t\rangle \mid s \text{ is blocked in } \mathcal{A}'' \text{ by } s' \text{ and } R(s',t) \in \mathcal{A}''\}
\end{aligned}$$

Due to (*), it is straightforward to see that $I \models \mathcal{A}''$. Consider now each HT-clause $r \in \mathcal{C}$ with a center variable x, and let σ be a mapping from the variables of r to individuals in $\mathcal{A}''$ such that $I \models \sigma(U_i)$ for each atom U_i from the antecedent of r. Let σ' be a mapping such that $\sigma'(y_i) = \sigma(y_i)$ for each branch variable; furthermore, $\sigma'(x) = \sigma(x)$ if $\sigma(x)$ is not blocked, and $\sigma'(x)$ is the blocker of $\sigma(x)$ otherwise. By $I \models \sigma(U_i)$ and the definition of I, we have that $\sigma'(U_i) \in \mathcal{A}''$. The *Hyp*-rule is not applicable to $\mathcal{A}''$ and $\mathcal{C}$ for σ', so $\sigma'(V_j) \in \mathcal{A}''$ for some atom in the consequent of r. The definitions of I and of single blocking then imply $I \models \sigma(V_j)$. $\square$

For DLs with inverse roles, single blocking must be applied with care in the hypertableau setting. Consider the following knowledge base $\mathcal{K}_{10}$, consisting of an ABox $\mathcal{A}_{10}$ and a set of DL-clauses $\mathcal{C}_{10}$.

$$\begin{aligned}
&\mathcal{A}_{10} = \{\ C(a)\ \} \\
&\mathcal{C}_{10} = \{C(x) \rightarrow \exists R.D(x),\ D(x) \rightarrow \exists S^-.C(x),\ R(x, y_1) \wedge S(x, y_2) \rightarrow \bot\}
\end{aligned} \tag{18}$$

On $\mathcal{K}_{10}$, the hypertableau algorithm produces the ABox shown in Figure 12. The individual c is single-blocked by a, so the algorithm terminates; an expansion of $\exists R.D(c)$, however, would reveal that $\mathcal{K}_{10}$ 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

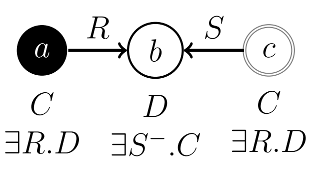


Fig. 12. Problems with Single Blocking

it to examine both the successors and the predecessor of x. Single 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 c is not blocked by a any more.

We can apply single blocking to the DL $\mathcal{SHIO}$ provided we assume that each DL-clause contains at most one role atom in the antecedent. We can always ensure this by suitably renaming complex concepts with atomic ones.

Lemma 20. *Let $\mathcal{A}$ be an ABox and $\mathcal{C}$ a set of HT-clauses such that, for each $r \in \mathcal{C}$, (*i*) r contains no atoms of the form $R(x, x)$, and (*ii*) the antecedent of r contains at most one role atom. If a derivation with single blocking for $\mathcal{C}$ and $\mathcal{A}$ exists in which a leaf node is labeled with a clash-free ABox $\mathcal{A}'$, then $(\mathcal{C}, \mathcal{A})$ is satisfiable.*

PROOF. Let $\mathcal{A}''$ be obtained from $\mathcal{A}'$ by removing all assertions containing an indirectly blocked individual. It is easy to see that no derivation rule is applicable to $\mathcal{A}''$ and $\mathcal{C}$. For an individual s occurring in $\mathcal{A}''$, let $[s]_{\mathcal{A}''} = s$ if s is not blocked in $\mathcal{A}''$, and let $[s]_{\mathcal{A}''} = s'$ if s' is blocked in $\mathcal{A}''$ by s. Given $\mathcal{A}''$, we construct an interpretation I as follows.

$$\begin{aligned}
\triangle^I &= \{s \mid s \text{ occurs in } \mathcal{A}'' \text{ and it is not blocked in } \mathcal{A}''\} \\
s^I &= [s]_{\mathcal{A}''} \text{ for each individual } s \text{ occurring in } \mathcal{A}'' \\
A^I &= \{[s]_{\mathcal{A}''} \mid A(s) \in \mathcal{A}''\} \\
R^I &= \{\langle [s]_{\mathcal{A}''}, [t]_{\mathcal{A}''} \rangle \mid R(s, t) \in \mathcal{A}''\}
\end{aligned}$$

It is straightforward to see that $I \models \mathcal{A}''$. Consider now each HT-clause $r \in \mathcal{C}$, and let σ be a mapping from the variables of r to the individuals in $\mathcal{A}''$ such that $I \models \sigma(U_i)$ for each atom U_i from the antecedent of r. If the antecedent of r does not contain role atoms, it is trivial to see that $I \models \sigma(V_j)$ for some atom V_j from the consequent of r. Assume now that the antecedent of r contains an atom of the form $R(x, y)$. By the definition of I, individuals s and t then exist such that $R(s, t) \in \mathcal{A}''$, $\sigma(x) = [s]_{\mathcal{A}''}$, and $\sigma(y) = [t]_{\mathcal{A}''}$. Let σ' be such that $\sigma'(x) = s$ and $\sigma'(y) = t$. Since the *Hyp*-rule is not applicable to $\mathcal{C}$ and $\mathcal{A}''$ for σ', we have $\sigma'(V_j) \in \mathcal{A}''$ for some atom V_j from the consequent of r. By the definitions of I and single blocking, we can then conclude that $I \models \sigma(V_j)$. □

5.3 The Number of Root Individuals

$\mathcal{SHOIQ}$ is NExpTime-complete [37], and it is straightforward to extend this proof to $\mathcal{SHOIQ}^+$. Thus, one might wonder whether the complexity result in Theorem 16 can be sharpened to obtain a worst-case optimal decision procedure. This, unfortunately, is not the case: we present an example on which our algorithm generates a doubly-exponential number of root individuals. We use the well-known encoding of binary numbers by concepts $B_0, B_1, \ldots, B_{k-1}$: we assign to each individual s in an ABox $\mathcal{A}$ a binary number $\ell_{\mathcal{A}}(s) = b_{k-1} \ldots b_1 b_0$ such that $b_i = 1$ if and only if $B_i(s) \in \mathcal{A}$. Thus, using k concepts, we can encode 2^k different binary numbers. Furthermore, for any atomic role R, using the well-known *R-successor counting formula* [37], we can ensure that, whenever an individual t is an R-successor of s in $\mathcal{A}$, then $\ell_{\mathcal{A}}(t) = (\ell_{\mathcal{A}}(s) + 1) \bmod 2^k$. We omit this formula for the sake of brevity and refer the interested reader to [37]. Let $\mathcal{K}_{11}$ be the knowledge base consisting of axioms (19)–(25). For the same reasons of brevity, we omit the DL-clauses corresponding to the axioms in $\mathcal{K}_{11}$.

(19) $$C(a)$$
(20) $$C \sqsubseteq \exists R.C \sqcap \exists L.C$$
(21) (*The R-successor formula for* $B_0, \ldots, B_{k-1}$)
(22) (*The L-successor formula for* $B_0, \ldots, B_{k-1}$)
(23) $$B_0 \sqcap \ldots \sqcap B_k \sqsubseteq \{b\}$$
(24) $$A \sqsubseteq \forall R^-.A \sqcap \forall L^-.A \sqcap \leq 1\, L^- \sqcap \leq 1\, R^-$$
(25) $$A(b)$$

On $\mathcal{K}_{11}$, our algorithm can exhibit a derivation, schematically shown in Figure 13, in which a doubly exponential number of root individuals is introduced. Due to (19)–(23), a sequence of 2^k individuals $a = x_0, x_1, \ldots, x_{2^k-1} = b$ connected by L can be generated. Then, because of (24)–(25), all of the individuals in the sequence can be labeled with A and turned into root individuals. But then, these individuals cannot be used as blockers, so the algorithm can create an R successor x'_1 of x_0, and then create a sequence of $2^k - 1$ individuals between x'_1 and b connected by L. Again, due to (24)–(25), all these individuals can be labeled with A and turned into root individuals, and the process can be repeated. Eventually, the algorithm can create a binary tree of root individuals of exponential depth, thus creating a doubly exponential number of root individuals in total.

5.4 The Number of Blockable Individuals

If $\mathcal{K}$ is a $\mathcal{SHIQ}$ knowledge base, then $\Xi(\mathcal{K})$ contains no nominal guard concepts, so the *NI*-rule can never be applied in a derivation for $\Xi(\mathcal{K})$. Thus, no new root individuals are introduced, which eliminates a source of complexity in our algorithm. Furthermore, in [11] the authors presented a tableau algorithm for the basic DL $\mathcal{ALC}$ which, due to anywhere blocking, runs in NExpTime instead of

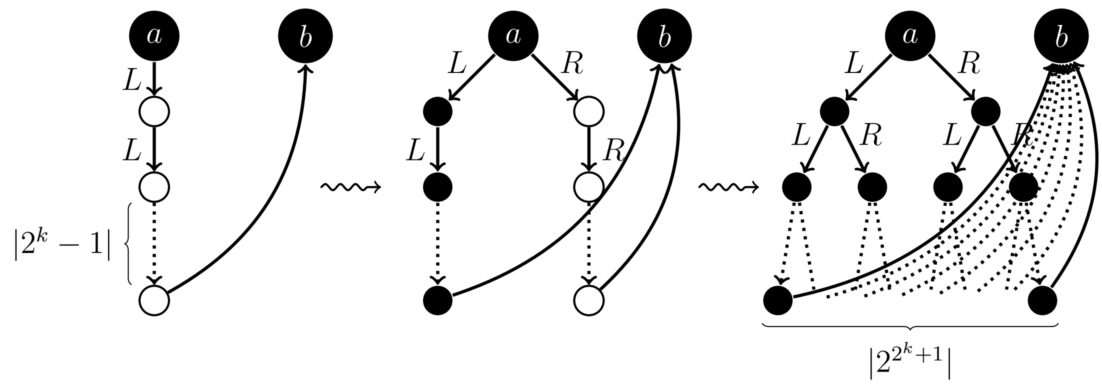


Fig. 13. Creation of an Exponentially Deep Binary Tree of Root Individuals

2NExpTime. It is therefore interesting to compare this algorithm to ours. Let $\mathcal{K}_{12}$ be the following knowledge base. Again, for the sake of brevity, we omit the DL-clauses corresponding to the axioms in $\mathcal{K}_{12}$.

(26) $C(a)$

(27) $C \sqsubseteq \exists L.C \sqcap \exists R.C$

(28) (*The R-successor formula for* $B_0, \ldots, B_{k-1}$)

(29) (*The L-successor formula for* $B_0, \ldots, B_{k-1}$)

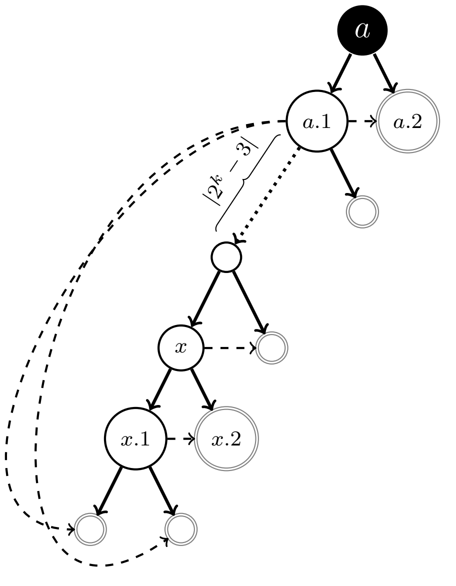
(30) $B_0 \sqcap \ldots \sqcap B_k \sqsubseteq A$

(31) $\exists L.A \sqcap \exists R.A \sqsubseteq A$

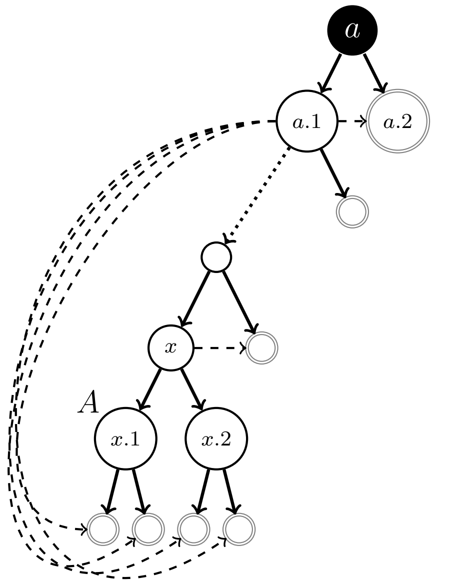
Figure 14 schematically presents a derivation on $\mathcal{K}_{12}$ in which a doubly exponential number of blockable individuals is introduced.[7] For the simplicity of presentation, we use single anywhere blocking. Due to (26)–(29), our algorithm can create individuals $a.1$, $a.2$, $a.1.1$, $a.1.2$, $a.1.1.1$, $a.1.1.2$, and so on, such that $s.1$ is an L-successor of s, and $s.2$ is an R-successor of s. After creating the individuals of the form $a.1^{2^k-1}.1$ and $a.1^{2^k-1}.2$ where 1^{2^k-1} is a string of 2^k-1 ones, each individual $x.1$ blocks $x.2$ (c.f. Figure 14a). But then, due to (30), $a.1^{2^k-1}.1$ and $a.1^{2^k-1}.2$ become instances of A. By (31), $a.1^{2^k-1}$ is made an instance of A as well, so it does not block its sibling $a.1^{2^k-2}.2$ any more; hence, $a.1^{2^k-2}.2$ is now expanded to the exponential depth (c.f. Figure 14b). By repeating the process, the algorithm derives that $a.1^{2^k-2}$ is an instance of A, but then it does not block its sibling $a.1^{2^k-3}.2$ (c.f. Figure 14d). By repeating the process, the algorithm constructs a binary tree of exponential depth, thus creating a doubly-exponential number of blockable nodes in total (c.f. Figure 14d).

The nondeterministic exponential behavior is obtained in [11] by applying the $\sqcap$-, $\sqcup$-, $\forall$-, and $\sqsubseteq$-rules exhaustively before applying the $\exists$-rule. Such a strategy en-

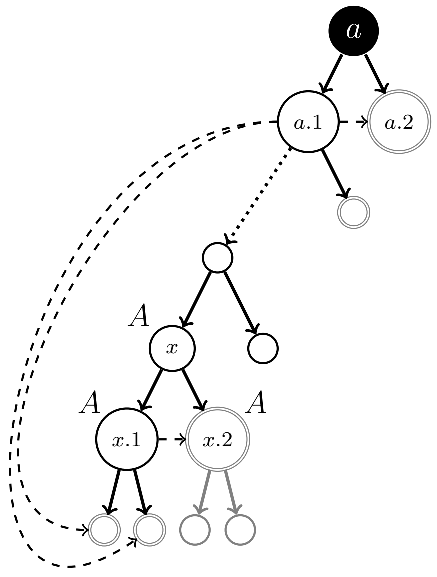
[7] In [30], we suggested informally that our algorithm should run in NExpTime on $\mathcal{SHIQ}$. As this example shows, this is not the case.



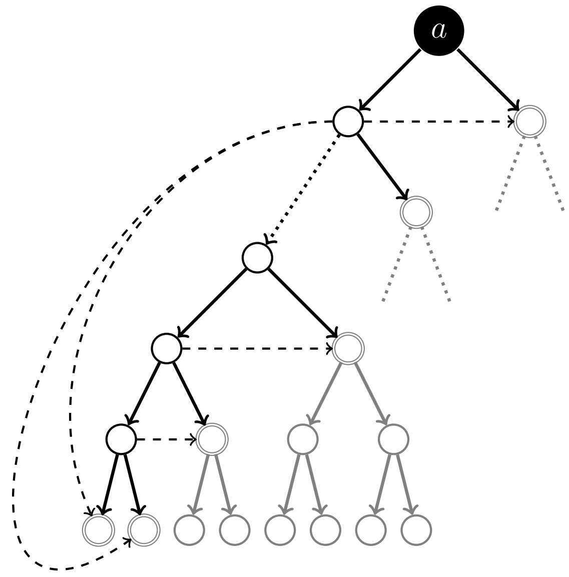
(a) An exponential path is constructed with each blockable individual blocking its sibling. No individual contains A in its label.



(b) Adding A to the label of $x.1$ unblocks $x.2$.



(c) Adding A to the label of $x.2$ makes $x.2$ blocked, and forces the addition of A to the label of x. This unblocks the sibling of x so another subtree is created.



(d) Derivation terminates with only exponentially many unblocked individuals, but with a doubly-exponential number of indirectly blocked individuals.

Fig. 14. Creation of an Exponentially Deep Binary Tree of Blockable Individuals

sures that the label of an individual s is fully constructed before any successors of s are introduced. On $\mathcal{K}_{12}$, this means that the GCI (31) is applied to each individual s *before* introducing its successors. Thus, before the existentials on s are expanded, the tableau algorithm from [11] introduces $(\forall L.\neg A \sqcup \forall R.\neg A \sqcup A)(s)$ and nondeterministically chooses one disjunct. The choices $(\forall L.\neg A)(s)$ and $(\forall R.\neg A)(s)$ will lead to a clash, so the algorithm eventually derives $A(s)$, before it expands the existentials on s and introduces $s.1$ and $s.2$. Thus, while generating only exponential models in the worst case, this algorithm incurs a massive amount of nondeterminism.

Nondeterministic exponential behavior can be guaranteed in our hypertableau algorithm by nondeterministically fixing the label of each individual before applying the $\leq$-rule to it. This technique is similar to the one used in [38] in order to obtain a PSpace decision procedure for concept satisfiability in a DL with inverse roles but without GCIs. The performance results in Section 7, however, seem to suggest that this might not be beneficial in practice. Still, it might be worth exploring whether nondeterministically adding concepts to labels of individuals can be used as an optimization that would detect "early blocks" and thus prevent the construction of large models.

6 Algorithm Optimizations

DL reasoning algorithms are often used in practice to compute a *classification* of a knowledge base $\mathcal{K}$—that is, to determine whether $\mathcal{K} \models A \sqsubseteq B$ for each pair of atomic concepts A and B occurring in $\mathcal{K}$. Clearly, a naïve classification algorithm would involve a quadratic number of calls to the subsumption checking algorithm, each of which can potentially be highly expensive. To obtain acceptable levels of performance, various optimizations have been developed that reduce the number of subsumption checks [4] and the time required for each check [3, Chapter 9]. Along these lines, we have developed two simple optimizations that, to the best of our knowledge, have not been considered previously in literature.

6.1 Reading Classification Relationships from Concept Labels

Let $(\mathcal{A}, \mathcal{C})$ be an ABox and a set of DL-clauses obtained by clausifying a knowledge base $\mathcal{K}$, and let A and B be atomic concepts for which we want to check whether $\mathcal{K} \models A \sqsubseteq B$; since A and B are atomic, this is the case if and only if $(\mathcal{A}', \mathcal{C})$ is unsatisfiable where $\mathcal{A}' = \mathcal{A} \cup \{A(a), \neg B(b)\}$ and a is a fresh individual. Let $\mathcal{A}_1$ be a clash-free ABox labeling a leaf in a derivation from $(\mathcal{A}', \mathcal{C})$. We can use $\mathcal{A}_1$ to learn the following things about subsumption in $\mathcal{K}$. The proof of these claims is trivial, and we omit if for the sake of brevity.

(1) If $C(a) \in \mathcal{A}_1$ and the derivation of $C(a)$ does not depend on a nondeterministic choice, then $\mathcal{K} \models A \sqsubseteq C$.

(2) If $\mathcal{A}_1$ has been obtained from $\mathcal{A}'$ deterministically, then $\mathcal{K} \models A \sqsubseteq C$ only if $C(a) \in \mathcal{A}_1$.
(3) If $C(b) \in \mathcal{A}_1$ but $D(b) \notin \mathcal{A}_1$ for C and D concepts and b an individual that is not blocked, then $\mathcal{K} \not\models C \sqsubseteq D$.

By the first two properties, if $\mathcal{K}$ is a deterministic knowledge base such as GALEN, we can classify it using a *linear* number of calls to the hypertableau algorithm: for each atomic concept A, we check satisfiability of $(\mathcal{A} \cup \{A(a)\}, \mathcal{C})$; if the algorithm produces a clash-free ABox $\mathcal{A}_1$, the superclasses of A are contained in $\mathcal{L}_{\mathcal{A}_1}(a)$. These optimizations are applicable in the case of tableau algorithms as well; however, due to increased or-branching, they are likely to be less effective.

6.2 Caching Blocking Labels

Let $\mathcal{T}$ and $\mathcal{R}$ be a $\mathcal{SHIQ}$ TBox and RBox, respectively, and let $\mathcal{C} = \Xi_{\mathcal{TR}}(\mathcal{T} \cup \mathcal{R})$. Furthermore, let us assume that the classification of $\mathcal{T} \cup \mathcal{R}$ involves n consecutive calls to the hypertableau algorithm for $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$. Then, if a derivation from $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$ contains a leaf node labeled with a clash-free ABox $\mathcal{A}_i$, we can use the nonblocked individuals from $\mathcal{A}_i$ as potential blockers in all subsequent satisfiability satisfiability checks of $(\{A_j(a_j), \neg B_j(a_j)\}, \mathcal{C})$ for $j > i$.

This is a simple consequence of the following fact. Let I_1 and I_2 be two models of $\mathcal{T} \cup \mathcal{R}$ such that $\triangle^{I_1} \cap \triangle^{I_2} = \emptyset$; furthermore, let I be defined as $\triangle^I = \triangle^{I_1} \cup \triangle^{I_2}$, $A^I = A^{I_1} \cup A^{I_2}$, and $R^I = R^{I_1} \cup R^{I_2}$, for each atomic concept A and each atomic role R. Then, by a simple induction on the structure of axioms in $\mathcal{T} \cup \mathcal{R}$, it is trivial to show that $I \models \mathcal{T} \cup \mathcal{R}$. This property does not hold in the presence of nominals, which can impose a bound the number of elements in a concept; the bound can be satisfied in I_1 and I_2 individually, but it can be violated in I.

Our optimization is correct because, instead of $(\{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$, we can check satisfiability of $(\mathcal{A}_i \cup \{A_i(a_i), \neg B_i(a_i)\}, \mathcal{C})$, and in doing so, we can use the individuals from $\mathcal{A}_i$ as potential blockers due to anywhere blocking. This optimization can be seen as a very simple form of model caching, and it has been key to obtaining the results that we present in Section 7. For example, on GALEN only one subsumption test is costly because it computes a substantial part of the model of the TBox; all subsequent subsumption tests reuse large parts of that model.

In practice, we do not need to keep entire ABox $\mathcal{A}_i$ around; rather, for each nonblocked blockable individual t with a predecessor t', we simply need to retain the sets $\mathcal{L}_{\mathcal{A}_i}(t)$, $\mathcal{L}_{\mathcal{A}_i}(t')$, $\mathcal{L}_{\mathcal{A}}(t,t)$, $\mathcal{L}_{\mathcal{A}_i}(t,t')$, and $\mathcal{L}_{\mathcal{A}_i}(t',t)$.

7 Implementation and Evaluation

Based on the calculus from Section 4, we have implemented a prototype DL reasoner called HermiT. The implementation fully supports reasoning with $\mathcal{SHOIQ}^+$ ontologies, and can perform both consistency and subsumption testing, as well as classification of atomic concepts in an ontology. An extensive discussion of the used implementation techniques is out of scope of this paper; we only briefly comment on the implementation of anywhere blocking.

In the DL community, it is common understanding that anywhere blocking is more costly than ancestor blocking because the set of possible blockers to be examined is potentially much larger. We developed, however, an efficient implementation of anywhere blocking. To determine the blocking status of all individuals in an ABox $\mathcal{A}$, our algorithm constructs a hash table in which individuals are indexed by their five blocking labels. The table is created by scanning all individuals in $\mathcal{A}$ in the increasing order of the precedence $\prec$. For each individual s in $\mathcal{A}$, if the parent of s is blocked, then s is indirectly blocked; otherwise, the algorithm tries to find in the hash table an individual whose five blocking labels are equal to the ones of s. If the hash table contains such an individual t, then s is directly blocked in $\mathcal{A}$ by t; otherwise, s is not blocked in $\mathcal{A}$ so it is added into the hash table. With a good hash table implementation, the blocking status of all individuals in $\mathcal{A}$ can be determined with a linear number of table lookups.

To evaluate our reasoning algorithm in practice, we compared the HermiT with state-of-the-art tableau reasoners Pellet 1.5.1 [31], and FaCT++ 1.1.10 [41]. Pellet and FaCT++ are based on the existing reasoning algorithms [21], so they differ from HermiT in the employed derivation rules and the ancestor blocking. In order to estimate the practical impact of these two differences separately, we implemented a version of HermiT with ancestor blocking, which we call HermiT-Anc.

We selected test ontologies from the Gardiner ontology suite [14]—a collection of OWL ontologies gathered from the Web; the Open Biological Ontologies (OBO) Foundry [8] —a collection of biology and life science ontologies; and the variants of the GALEN ontology [33]—a large and complex biomedical ontology. Most ontologies from the Gardiner and OBO collections contain datatypes, which are currently not supported in HermiT; therefore, we have converted datatypes in these ontologies to atomic classes. After eliminating syntactically incorrect OWL ontologies from, we obtained a test suite consisting of 139 ontologies.

We measured the time needed to classify each test ontology using all of the mentioned reasoners. All tests were performed on a 2.2 GHz MacBook Pro with 2 GB of physical memory. A classification attempt was aborted if it exhausted all available memory (Java tools were allowed to use 1.5 GB of heap space), or if it exceeded a timeout of 20 minutes.

The majority of the test ontologies—126 of them, to be precise—were classified in under a second by HermiT, and under ten seconds by Pellet and FaCT++. For these "trivial" ontologies, the performance of HermiT was comparable to that of all other reasoners. Therefore, we discuss here only the tests results for "interesting" ontologies—that is, ontologies that are either not trivial or on which the tested reasoners exhibited a significant difference in performance.

Table 9 lists these "interesting" ontologies and presents the statistical information about their contents. These ontologies include large biomedical ontologies,

[8] `http://obofoundry.org/`

Table 9
Statistics of Test Ontologies

			Numbers of Axioms		
Ontology Name	Classes	Individuals	TBox	ABox	RBox
Fly Taxonomy	6,602	5,350	6,603	5,350	0
GO Term DB	20,526	0	28,997	0	1
Biological Process	14,955	73,901	27,051	73,901	1
NCI	27,652	0	46,940	0	0
MGED	216	579	430	642	0
BP XP OBOL	10,295	43,446	6,980	43,446	0
OWL Guide Food	139	207	386	453	9
FMA Lite	75,141	46,225	119,558	46,225	3
DLP ExtDnS	96	0	606	0	384
FMA-constitutional part	41,651	93	123,061	1	57
GALEN-horrocks	2,748	0	4,087	0	442
Not-GALEN	3,087	0	5,329	0	442
GALEN-doctored	2,748	0	4,087	0	649
GALEN-original	2,748	0	4,330	0	649
GALEN-module1	6,362	0	14,574	0	209
GALEN-full	23,136	0	36,205	0	1,966

such as several ontologies from the OBO corpus (Fly Taxonomy, Biological Process, and BP XP OBOL), a version of the Gene Ontology (GO Term DB) [1], the National Cancer Institute Thesaurus (NCI) [19], and two versions of the Foundational Model of Anatomy (FMA Lite and FMA-constitutional part) [16]. The Food ontology from the OWL Guide[9] is fairly small, but it makes heavy use of all features of $\mathcal{SHOIN}$.

Table 10 summarizes the results of tests on the "interesting" ontologies. In most cases, HermiT performs as well as or better than the other reasoners. Because HermiT can perform most of its reasoning deterministically, it can apply the optimization described in Section 6.1 and thus reduce the total number of subsumption tests. Furthermore, HermiT's classification time is in most cases dominated by only the first such satisfiability test, as the caching of blocking labels described in Section 6.2 makes subsequent tests easy.

HermiT performs worse than Pellet and FaCT++ on the DLP ExtDnS ontology. This ontology includes a substantially more complex RBox than most other ontologies in the test suite, with 384 role axioms. Our analysis revealed that HermiT's poor performance on DLP ExtDnS is due to the large number of GCIs introduced by the encoding of transitivity described in Section 4.1.1. Consider the following

[9] `http://www.w3.org/TR/owl-guide/`

Table 10
Results of Performance Evaluation

Ontology Name	Classification Times (seconds)			
	HermiT	HermiT-Anc	Pellet	FaCT++
Fly Taxonomy	1.1	1.2	1.2	5.3
GO Term DB	1.6	1.8	36.4	19.2
Biological Process	2.4	1.6	10.7	79.2
NCI	2.8	3.7	17.0	30.2
MGED	5.7	11.2	0.8	0.249
BP XP OBOL	8.7	8.5	505.1	1742.3
OWL Guide Food	19.3	29.6	14.2	1388.1
FMA Lite	43.8	error	error	error
DLP ExtDnS	95.8	error	7.1	0.1
FMA-constitutional part	error	error	error	error
GALEN-horrocks	1.5	1.5	13.5	156.9
Not-GALEN	1.6	1.8	54.1	200.4
GALEN-doctored	3.9	4.9	error	2836.1
GALEN-original	11.9	error	error	error
GALEN-module1	error	error	error	error
GALEN-full	error	error	error	error

example knowledge base, which abstracts the RBox of DLP ExtDnS.

$$\mathcal{K}_{13} = \left\{ \begin{array}{l} R_1 \sqsubseteq R_2,\ \ R_2 \sqsubseteq R_3,\ \ \ldots,\ \ R_{m-1} \sqsubseteq R_m, \\ \mathsf{Tra}(R_1),\ \ \mathsf{Tra}(R_2),\ \ \ldots,\ \ \mathsf{Tra}(R_m), \\ A \sqsubseteq \forall R_m.B_1 \sqcup \forall R_m.B_2 \sqcup \ldots \sqcup \forall R_m.B_n \end{array} \right\} \tag{32}$$

The concept closure $\mathsf{clos}(\mathcal{K}_{13})$ contains a concept $\forall R_i.B_j$ for each $1 \leq i \leq m$ and $1 \leq j \leq n$, and the encoding $\Omega(\mathcal{K}_{13})$ contains an axiom $\forall R_i.B_j \sqsubseteq \forall R_k.(\forall R_k.B_j)$ for each i, j, and k such that $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq k \leq i$. The number of axioms introduced by the transitivity encoding can thus be cubic in the size of the overall knowledge base (and quadratic in the size of the RBox). On DLP ExtDnS, these additional axioms lead to inefficiency. In contrast, the $\forall_+$-rule used in standard tableau algorithm does not suffer from these problems.

HermiT also performs worse on than Pellet and FaCT++ on the MGED ontology. This ontology contains nominals, as well as a moderately complex ABox (over 600 assertions). Since the ontology uses nominals, the ABox must be taken into account when classifying the ontology. The *completion graph caching* optimization [36] can be used to avoid repeating all same reasoning steps over the ABox for each independent subsumption test; however, HermiT does not implement this optimization yet, which may explain its worse performance. Furthermore, as explained in Section 6.2, the presence of nominals prevents us from applying the caching of blocking labels. HermiT completes each subsumption test quite quickly (on the order of 30ms), suggesting that HermiT should achieve performance levels comparable to other reasoners once completion graph caching is implemented.

Different versions of GALEN have commonly been used for testing performance of DL reasoners. The full version of the ontology (called GALEN-full) cannot be processed by any of the reasoners. To simplify the ontology, we extracted a module (called GALEN-module1) from GALEN-full using the techniques from [17]. Although the module is much smaller than the full ontology, no reasoner was able to classify it either. Similarly, no reasoner could classify FMA-constitutional part. Our analysis has shown that, due to a large number of cyclic axioms, on these ontologies reasoners construct extremely large ABoxes and eventually exhaust all available memory. To alleviate this problem, e are currently developing a reasoning technique in which the $\exists$-rule is modified such that it nondeterministically tries to reuse individuals form the ABox generated thus far. Our initial experiments have shown very promising results [29].

Many simplified versions of GALEN have often been used in practice. GALEN-horrocks and Not-GALEN are the versions found in the Gardiner suite; furthermore, GALEN-original is the version of GALEN from roughly 10 years ago, and GALEN-doctored has been obtained from GALEN-original by removing about a 100 "difficult" axioms. As Table 10 shows, these ontologies are still challenging for the existing reasoners. HermiT, however, can classify them quite efficiently; in fact, HermiT is the only reasoner that can classify GALEN-original. All other reasoners, as well as HermiT-Anc, quickly run our of memory on GALEN-original; this suggests that, by drastically reducing sizes of generated ABoxes, anywhere blocking can mean the difference between success and failure on complex ontologies.

On ontologies that can be processed by both HermiT and HermiT-Anc, both reasoners show comparable performance, suggesting that the ABoxes generated on these ontologies are not particularly large. On some of these ontologies (e.g., BP XP OBOL and OWL Guide Food), Pellet and FaCT++ perform significantly slower; this suggests that the increase in HermiT's performance is mainly due to the hypertableau rule application strategy and the reduced nondeterminism. Thus, while the hypertableau strategy may not be as important as anywhere blocking in determining the practical limits of DL reasoners, it can lead to significant performance improvements in practice.

8 Conclusion

In this paper, we presented a novel reasoning algorithm for DLs. The algorithm is based on the hyperresolution inferences and anywhere blocking, which reduces the nondeterminism due to GCIs and the sizes of the generated models. Furthermore, the algorithm uses a novel approach to handling the interaction between nominals, inverse roles, and number restrictions, which seems to be easier to implement in practice than the existing techniques [21]. We have implemented our calculus and have conducted an extensive performance comparison. Our results show that the combination of novel optimizations significantly increase the performance of DL reasoning in practice: our reasoner is currently the only one that can handle certain versions of GALEN—a medical terminology that has proven notoriously difficult for DL reasoners.

To further improve the performance of DL reasoning, we shall investigate optimizations that might help us reduce sizes of the generated models. Furthermore, we shall try to extend our technique to the DL $\mathcal{SROIQ}$, which underpins the OWL 1.1—the extension of OWL currently being standardized by the W3C.

References

[1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontolgy: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29, 2000.

[2] F. Baader, M. Buchheit, and B. Hollunder. Cardinality Restrictions on Concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[3] 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, 2nd edition, August 2007.

[4] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An Emperical Analysis of Optimization Techniques for Terminological Representation systems or: "Making KRIS Get a Move on". *Applied Intelligence*, 4(2):109–132, 1994.

[5] F. Baader and U. Sattler. An Overview of Tableau Algorithms for Description Logics. *Studia Logica*, 69:5–40, 2001.

[6] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. of the European Workshop on Logics in Artificial Intelligence (JELIA '96)*, number 1126 in LNAI, pages 1–17, Évora, Portugal, September 30–October 3 1996. Springer.

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

[8] F. Bry and S. Torge. A Deduction Method Complete for Refutation and Finite Satisfiability. In J. Dix, L. F. del Cerro, and U. Furbach, editors, *Proc. European Workshop on Logics in Artificial Intelligence (JELIA '98)*, volume 1489 of *LNCS*, pages 122–138, Dagstuhl, Germany, October 12–15 1998. Springer.

[9] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[10] S. Demri and H. de Nivelle. Deciding Regular Grammar Logics with Converse Through First-Order Logic. *Journal of Logic, Language and Information*, 14(3):289–329, 2005.

[11] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[12] C. Fermüller, T. Tammet, N. Zamov, and A. Leitsch. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI*. Springer, 1993.

[13] 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.

[14] T. Gardiner, I. Horrocks, and D. Tsarkov. Automated Benchmarking of Description Logic Reasoners. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, 2006.

[15] L. Georgieva, U. Hustadt R. A., and Schmidt. Hyperresolution for Guarded Formulae. *Journal of Symbolic Computation*, 36(1–2):163–192, 2003.

[16] C. Golbreich, S. Zhang, and O. Bodenreider. The Foundational Model of Anatomy in OWL: Experience and Perspectives. *Journal of Web Semantics*, 4(3):181–195, 2006.

[17] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Modular Reuse of Ontologies: Theory and Practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.

[18] V. Haarslev and R. Möller. RACER System Description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the 1st Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *LNAI*, pages 701–706, Siena, Italy, June 18–23 2001. Springer.

[19] F. W. Hartel, S. de Coronado, R. Dionne, G. Fragoso, and J. Golbeck. Modeling a description logic vocabulary for cancer research. *Journal of Biomedical Informatics*, 38(2):114–129, 2005.

[20] I. Horrocks and U. Sattler. Ontology Reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ Description Logic. In B. Nebel, editor, *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, Seattle, WA, USA, August 4–10 2001. Morgan Kaufmann Publishers.

[21] I. Horrocks and U. Sattler. A Tableau Decision Procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007.

[22] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[23] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In D. MacAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, volume 1831 of *LNAI*, pages 482–496, Pittsburgh, USA, June 17–20 2000. Springer.

[24] A. K. Hudek and G. Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In B. Parsia, U. Sattler, and D. Toman, editors, *Proc. of the 2006 Int. Workshop on Description Logics (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, Windermere, UK, May 30-June 1 2006.

[25] U. Hustadt, B. Motik, and U. Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[26] U. Hustadt and R. A. Schmidt. Issues of Decidability for Description Logics in the Framework of Resolution. In R. Caferra and G. Salzer, editors, *Selected Papers from Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 191–205. Springer, 1999.

[27] O. Kutz, I. Horrocks, and U. Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[28] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.

[29] B. Motik and I. Horrocks. Individual Reuse in Description Logic Reasoning. In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR 2008)*, Sydney, Australia, August 10–15 2008. Springer. To appear.

[30] B. Motik, R. Shearer, and I. Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In F. Pfenning, editor, *Proc. of the 21st Conference on Automated Deduction (CADE-21)*, volume 4603 of *LNAI*, pages 67–83, Bremen, Germany, July 17–20 2007. Springer.

[31] B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner. Poster, In Proc. of the 3rd Int. Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 7–11, 2004.

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

[33] A. L. Rector and J. Rogers. Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN. In P. Barahona, F. Bry, E. Franconi, N. Henze, and U. Sattler, editors, *Tutorial Lectures of the 2nd Int. Summer School 2006*, volume 4126 of *LNCS*, pages 197–231, Lisbon, Portugal, September 4–8 2006. Springer.

[34] A. Robinson. Automatic Deduction with Hyper-Resolution. *Int. Journal of Computer Mathematics*, 1:227–234, 1965.

[35] R. A. Schmidt and U. Hustadt. A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae. In F. Baader, editor, *Proc. of the 19th Int. Conf. on Automated Deduction (CADE-19)*, volume 2741 of *LNAI*, pages 412–426, Miami Beach, FL, USA, July 28–August 2 2003. Springer.

[36] E. Sirin, B. Cuenca Grau, and B. Parsia. From Wine to Water: Optimizing Description Logic Reasoning for Nominals. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 90–99, Lake District, UK, June 2–5 2006. AAAI Press.

[37] S. Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[38] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen, Germany, 2001.

[39] D. Tsarkov and I. Horrocks. Efficient Reasoning with Range and Domain Constraints. In V. Haarslev and R. Möller, editors, *Proc. of the 2004 Int. Workshop on Description Logics (DL 2004)*, volume 104 of *CEUR Workshop Proceedings*, Whistler, BC, Canada, June 6–8 2004.

[40] D. Tsarkov and I. Horrocks. Ordering Heuristics for Description Logic Reasoning. In L. Pack Kaelbling and A. Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 609–614, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[41] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 292–297, Seattle, WA, USA, August 17–20 2006. Springer.