Acta Cybernetica **19** (2010) 571–588.

A Resolution Based Description Logic Calculus

Zsolt Zombori[*]

Abstract

We present a resolution based reasoning algorithm called *DL calculus* that decides concept satisfiability for the $\mathcal{SHQ}$ language. Unlike existing resolution based approaches, the DL calculus is defined directly on DL expressions. We argue that working on this high level of abstraction provides an easier to grasp algorithm with less intermediary transformation steps and increased efficiency. We give a proof of the completeness of our algorithm that relies solely on the $\mathcal{ALCHQ}$ tableau method, without requiring any further background knowledge.

1 Introduction and background

The Tableau Method [1] has long provided the theoretical background for DL reasoning and most existing DL reasoners implement some of its numerous variants. The typical DL reasoning tasks can be reduced to consistency checking and this is exactly what the Tableau Method provides. While the Tableau itself has proven to be very efficient, the reduction to consistency check is rather costly for some reasoning tasks. In particular, the ABox reasoning task *instance retrieval* requires running the Tableau Method for every single individual that appears in the knowledge base. Several techniques have been developed to make tableau-based reasoning more efficient on large data sets, (see e.g. [4]), that are used by the state-of-the-art DL reasoners, such as RacerPro [5] or Pellet [11].

Other approaches use first-order resolution for reasoning. A resolution-based inference algorithm is described in [7] which is not as sensitive to the increase of the ABox size as the tableau-based methods. The system KAON2 [10] is an implementation of this approach, providing reasoning services over the description logic language $\mathcal{SHIQ}$. The algorithm used in KAON2 in itself is not any more efficient for instance retrieval than the Tableau, but several steps that involve only the TBox can be performed before accessing the ABox, after which some axioms can be eliminated because they play no further role in the reasoning. This yields a qualitatively simpler set of axioms which then can be used for an efficient, query driven data reasoning. For the second phase of reasoning KAON2 uses a disjunctive datalog engine and not the original calculus. Thanks to the preprocessing, query

[*]Budapest University of Technology and Economics, Department of Computer Science and Information Theory, E-mail: zombori@cs.bme.hu

answering is very focused, i.e., it accesses as little part of the ABox as possible. However, in order for this to work, KAON2 still needs to go through the whole ABox once at the end of the first phase.

Reading the whole ABox is not a feasible option in case the ABox is bigger than the available memory or the content of the ABox changes so frequently that on-the-fly ABox access is an utmost necessity. Typical such scenarios include reasoning on web-scale or using description logic ontologies directly on top of existing information sources, such as in a DL based information integration system.

We have developed a DL ABox reasoner called DLog [9], available to download at `http://dlog-reasoner.sourceforge.net`, which is built on similar principles to KAON2. We will only highlight two main differences. First, instead of a datalog engine, we use the reasoning mechanism of the Prolog language [3] to perform the second phase (see [8]). Second, we use a modified resolution calculus (see [12]) that allows us to perform more inference steps in the first phase, thanks to which more axioms can be eliminated, yielding an even simpler set of axioms to work with in the second phase. The important difference is that while the approach of [10] can only guarantee that there are no nested functional symbols, our calculus ensures that no function symbols remain at all. This makes the subsequent reasoning easier and we can perform focused, query driven reasoning without any transformation that would require going through the ABox even once.

[12] describes the first phase of the reasoning algorithm implmented in DLog. The DL calculus presented in [13] aims to improve on this algorithm. We move the resolution-based reasoning from the level of first-order clauses to DL axioms, which saves us many intermediary transformation steps. Our current paper is the revised and corrected version of [13]. To avoid a problem in the proof published in [13], we had to restrict the calculus from the $\mathcal{SHIQ}$ language to the $\mathcal{SHQ}$ language. We hope to lift this restriction in the near future.

Our work is yet incomplete in that we only provide an algorithm for TBox reasoning. Although we sketch an extension to ABox reasoning at the end of the paper, we do not yet have a proof for its correctness.

This paper is structured as follows. First, in Section 2 we give a brief introduction to Description Logics and in particular the $\mathcal{SHQ}$ language. In Section 3 we present the DL calculus that performs consistency check for a $\mathcal{SHQ}$ TBox, and show that it can also be used to decide concept satisfiability. In Section 4 we discuss the time complexity of the algorithm. In Section 5 we prove the soundness of the DL calculus. In Section 6 we prove that the calculus is complete. Section 7 introduces our future work, in particular the extension of the DL calculus to ABox reasoning. Finally, Section 8 concludes by giving a brief summary of our results.

2 Description Logic

Description Logics (DLs) is a family of simple logic languages used for knowledge representation (for a detailed introduction see [1]). The language expressions use two main building blocks: *atomic concepts* that represent sets of objects and *atomic* *roles* that are used to describe relations between objects and stand for sets of object pairs. These building blocks can be combined to create *composite concepts* – as well as *composite roles* for some DL variants.

2.1 Terminological Axioms and Assertions

A DL statement can be an assertion about concrete individuals or it can express some general knowledge, very much like a rule. Statements of the first kind are called *data assertions* that are altogether referred to as the *Assertion box*, or *ABox*. Rule-like statements are called *terminology axioms* that constitute the *Terminology box*, or *TBox*.

2.2 The $\mathcal{SHQ}$ language

All that has been said so far is true of all DL languages. The members of the language family differ in the constructors that are available for building composite concepts and roles. We will be concerned with the $\mathcal{SHQ}$ language and give a brief summary to its syntax.

Consider a set N_C of atomic concepts, N_R of atomic roles and finitely many constant names. Using nothing but these, we can already make simple assertions. We can describe that an individual satisfies some atomic concept ($A(a)$), some atomic role holds between two individuals ($R(a, b)$), two individuals are equal ($a = b$) or they are distinct ($a \neq b$). We can declare some role to be transitive (Trans(R)). We can force a role to be subsumed by another ($R_1 \sqsubseteq R_2$), i.e., that every object pair that satisfies the first role also satisfies the second.

Let $\sqsubseteq^*$ the reflexive-transitive closure of the $\sqsubseteq$ relation. A role R is said to be *simple* if there is no role S such that Trans(S) and $S \sqsubseteq^* R$ hold.

There are no role constructors in the $\mathcal{SHQ}$ language. However, the following concept constructions are available:

A	Atomic concept, $A \in N_C$
$\top$	top (universal concept)
$\bot$	bottom (empty concept)
$\neg C$	complement of C
$C_1 \sqcap C_2$	intersection of C_1 and C_2
$C_1 \sqcup C_2$	union of C_1 and C_2
$\forall R.C$	value restriction (R, C arbitrary role and concept)
$\exists R.C$	existential resctriction (R, C arbitrary role and concept)
$\leq nS.C$	at-most number restriction (S simple role, C arbitrary concept)
$\geq nS.C$	at-least number restriction (S simple role, C arbitrary concept)

Two arbitrary concepts (simple or composite) can be asserted to be equivalent ($C \equiv D$), or that the one is subsumed by the other ($C \sqsubseteq D$). In summary, we give the valid statements of the $\mathcal{SHQ}$ language in Table 1.

Table 1: $\mathcal{SHQ}$ axioms

$\mathcal{SHQ}$ TBox	
$C \sqsubseteq D$	C, D arbitrary concepts
$C \equiv D$	C, D arbitrary concepts
$R \sqsubseteq S$	R, S arbitrary roles
Trans(R)	R arbitrary role
$\mathcal{SHQ}$ ABox	
$C(a)$	C arbitrary concept
$R(a, b)$	R arbitrary role
$a = b$	
$a \neq b$	

3 The DL Calculus

In this section we present a reasoning algorithm, called *DL calculus*, which decides the consistency of a $\mathcal{SHQ}$ TBox. Evidently, such an algorithm can be used for deciding concept satisfiability as well. To see this, suppose we want to know whether concept C is satisfiable in the presence of TBox $\mathcal{T}$. Take a role R that appears neither in $\mathcal{T}$ nor in C. Let us consider a new TBox $\mathcal{T}' = \mathcal{T} \cup \{\top \sqsubseteq \exists R.C\}$. Given that R is a new role name, it is easy to see that the newly added axiom will only introduce inconsistency to the TBox if C is unsatisfiable. C is satisfiable in the presence of TBox $\mathcal{T}$ if and only if $\mathcal{T}'$ is consistent. Hence, by giving an algorithm for TBox consistency check, we also provide an algorithm for concept satisfiability check.

The algorithm can be summarized as follows. We determine a set of concepts that have to be satisfied by each individual of an interpretation in order for the TBox to be true. Next, we introduce inference rules that derive a new concept from two concepts. Using the inference rules, we saturate the knowledge base, i.e., we apply the rules as long as possible and add the consequent to the knowledge base. We also apply redundancy elimination: whenever a concept extends another, it can be safely eliminated from the knowledge base [2]. It can be shown that saturation terminates. We claim that the knowledge base is inconsistent if and only if the saturated set contains the empty concept ($\bot$).

3.1 Preprocessing

We first eliminate transitivity from the knowledge base. It can be shown (see [10]) that any $\mathcal{SHQ}$ knowledge base KB can be transformed into a knowledge base KB' that contains no transitivity axioms and KB' is satisfiable if and only if KB is satisfiable.

Next, we internalize the TBox, i.e., we transform all axioms into a set of concepts that have to be satisfied by each individual. For instance, the axiom $C \sqsubseteq D$ is equivalent to the axiom $\top \sqsubseteq \neg C \sqcup D$, which amounts to saying that $\neg C \sqcup D$ has to be satisfied by all individuals.

Internalization is followed by structural transformation which eliminates the nesting of composite concepts into each other. A $\mathcal{SHQ}$ expression that appears in the TBox can be of arbitrary complexity, i.e., all sorts of composite concepts can appear within another concept. This makes reasoning very difficult. To solve this problem, we eliminate nesting composite concepts into each other by introducing new concept symbols that serve as names for embedded concepts. For details, see [10].

Finally, we make a small syntactic transformation: concepts $\forall R.C$ and $\exists R.D$ are replaced with equivalent concepts $(\leq 0R.\neg C)$ and $(\geq 1R.D)$, respectively. As a result, we obtain the following types of concepts, where L is a possibly negated atomic concept and R an arbitrary role:

$$\begin{array}{l} L_1 \sqcup L_2 \sqcup \cdots \sqcup L_i \\ L_1 \sqcup (\geq kR.L_2) \\ L_1 \sqcup (\leq nR.L_2) \end{array}$$

3.2 Notation

Before presenting the inference rules, we define some important notions. A *literal concept* (typically denoted with L) is a possibly negated atomic concept. A *bool concept* contains no role expressions (allowing only negation, union and intersection). We use capital letters from the beginning of the alphabet $(A, B, C \dots)$ to refer to bool concepts. In the following, we will always assume that a bool concept is presented in a simplest disjunctive normal form, i.e., it is the disjunction of conjunctions of literal concepts. So for example, instead of $A \sqcup A \sqcup (B \sqcap \neg B \sqcap C)$ we write A, and $A \sqcap \neg A$ is replaced with $\bot$. To achieve this, we apply eagerly the simplification rules presented in Figure 2 (see Subsection 3.5). When the inference rules (see Figure 1) do not preserve disjunctive normal form (DNF), we will use the explicit *dnf* operator:

$$dnf(A \sqcap B) = \begin{cases} dnf(A_1 \sqcap B) \sqcup dnf(A_2 \sqcap B) & \text{if } A = A_1 \sqcup A_2 \\ dnf(A \sqcap B_1) \sqcup dnf(A \sqcap B_2) & \text{if } B = B_1 \sqcup B_2 \\ (A \sqcap B) & \text{otherwise} \end{cases}$$

The *dnf* operator is defined only for concepts that are the intersection of two concepts. The bool concepts in the premises are always in DNF and the conclusion contains either the union or the intersection of such concepts. The union of two DNF concepts is also in DNF so we only need to apply the *dnf* operator to transform the intersection of two DNF concepts.

3.3 Ordering

Let $\succ$ be a total ordering, called a *precedence*, on the set of (atomic concept, atomic role, natural number, logic) symbols, such that $\geq \succ \leq \succ R \succ n \succ C \succ \neg \succ \sqcup \succ$

$\sqcap \succ \top \succ \bot$ for any atomic concept C, atomic role name R and natural number n; furthermore for any two natural numbers $n_1 \succ n_2$ if and only if $n_1 > n_2$. We define a corresponding *lexicographic path ordering* $\succ_{lpo}$ (see [2]) as follows:

$s = f(s_1, \ldots, s_m) \succ_{lpo} g(t_1, \ldots, t_n) = t$ if and only if

1. $f \succ g$ and $s \succ_{lpo} t_i$, for all i with $1 \leq i \leq n$; or
2. $f = g$ and, for some j, we have $(s_1, \ldots, s_{j-1}) = (t_1, \ldots, t_{j-1}), s_j \succ_{lpo} t_j$, and $s \succ_{lpo} t_k$, for all k with $j < k \leq n$; or
3. $s_j \succeq_{lpo} t$, for some j with $1 \leq j \leq m$.

In order for the above definition to be applicable, we treat concept $(\geq kS.A)$ as $\geq(k, S, A)$ and concept $(\leq nR.D)$ as $\leq(n, R, D)$. If the precedence is total on the symbols of the language, then the lexicographic path ordering is total on DL expressions. For simplicity, we often write $\succ$ instead of $\succ_{lpo}$ when it does not lead to confusion. Note a couple properties of our ordering that will be useful later:

1. A $\geq$-concept is greater than any $\leq$-concept or any bool concept.
2. A $\leq$-concept is greater than any bool concept.
3. $C_1 = (\leq n_1R_1.A_1)$ is greater than $C_2 = (\leq n_2R_2.A_2)$ if and only if:
   - $R_1 \succ R_2$ or
   - $R_1 = R_2$ and $n_1 > n_2$ or
   - $R_1 = R_2$, $n_1 = n_2$ and $A_1 \succ A_2$

Definition 1 (maximal concept)**.** *Given a set N of concepts, concept $C \in N$ is* maximal *in N if C is greater than any other concept in N.*

Since the ordering $\succ_{lpo}$ is total, for any finite set N there is always a unique concept $C \in N$ that is maximal in N.

3.4 $\mathcal{SHQ}$-concepts

A derivation in the DL calculus generates concepts that are more general than the ones obtained after preprocessing (see Subsection 3.1). We call this broader set $\mathcal{SHQ}$*-concepts*, defined as follows (C, D, E stand for concepts containing no role expressions):

$$C \qquad \text{(bool concepts)}$$

$$C \sqcup \bigsqcup(\leq nR.D) \qquad (\leq \text{-max concepts})$$

$$C \sqcup \left(\bigsqcup(\leq nR.D)\right) \sqcup (\geq kS.E) \qquad (\geq \text{-max concepts})$$

where bool concepts C, D, E are in DNF. Note two important properties of $\mathcal{SHQ}$-concepts:

1. A $\mathcal{SHQ}$-concept is a disjunction that contains at most one $\geq$-concept.
2. There are no nested concepts containing role expressions, i.e., a concept embedded into a $\geq$-concept or a $\leq$-concept is always a bool concept.

According to the ordering defined in Subsection 3.3, each $\leq$-concept is greater than any bool concept, so the maximal disjunct in a $\leq$-max concept is a $\leq$-concept. Similarly, any $\geq$-concept is greater than any $\leq$- or bool concept, so the maximal disjunct in a $\geq$-max concept is a $\geq$-concept. This is the rationale for naming these concepts $\leq$-max and $\geq$-max, respectively.

Obviously, any concept obtained after preprocessing is a $\mathcal{SHQ}$-concept:

Proposition 1. *For any $\mathcal{SHQ}$ knowledge base KB, if we apply the transformations described in Subsection 3.1 on KB, we obtain a set of $\mathcal{SHQ}$-concepts.*

3.5 Inference Rules

The inference rules are presented in Figure 1, where C_i, D_i, E_i are possibly empty bool concepts. W_i stands for an arbitrary $\mathcal{SHQ}$-concept that can be empty as well. Some of the rules do not preserve the disjunctive normal form (DNF) of bool concepts. In such cases, we use the *dnf* operator as defined in Subsection 3.2. Note that two disjunctive concepts are resolved along their respective maximal disjuncts and the ordering that we imposed on the concepts yields a selection function. Since the odering is total, we can always select the unique maximal disjunct to perform the inference step.

Along with the inference rules, we use a futher set of rules that we call *simplification rules* and which are shown in Figure 2. These rules only have one premise which is redundant in the presence of the conclusion and hence can be eliminated. In other words, the simplification rules are used to simplify concepts and do not deduce new concepts. Simplification rules are applied not only to $\mathcal{SHQ}$-concepts, but also to subconcepts appearing in $\mathcal{SHQ}$-concepts. For example, S1 is used to replace the concept $C \sqcup A \sqcup A$ with $C \sqcup A$, but also to replace $(\geq nR.(C \sqcup A \sqcup A))$ with $(\geq nR.(C \sqcup A))$.

Rule1 corresponds to the classical resolution inference and Rule2 makes this same inference possible for entities whose existence is required by $\geq$-concepts. Rule3 and Rule4 are harder to understand. They address the interaction between $\geq$-concepts and $\leq$-concepts. Intuitively, if some entity satisfies $\leq nR.C$ and also satisfies $\geq kS.D$, then there is a potential for clash if concepts C and D are related, more precisely if D is subsumed by C. In such cases $D \sqcap \neg C$ is not satisfiable, which either leads to contradiction if $n < k$ (Rule3) or results in a tighter cardinality restriction on the entity (Rule4). If several $\geq$-concepts and a $\leq$-concept are inconsistent together, then each $\geq$-concept is used to deduce a $\leq$-concept with smaller cardinality (Rule4) until the $\leq$-concept completely disappears from the conclusion (Rule3) and we obtain the empty concept.

Rule1 $$\frac{C_1 \sqcup (D_1 \sqcap A) \qquad C_2 \sqcup (D_2 \sqcap \neg A)}{C_1 \sqcup C_2}$$

where $D_1 \sqcap A$ is maximal in $C_1 \sqcup (D_1 \sqcap A)$
and $D_2 \sqcap \neg A$ is maximal in $C_2 \sqcup (D_2 \sqcap \neg A)$

Rule2 $$\frac{C \qquad W \sqcup (\geq nR.D)}{W \sqcup (\geq nR.dnf(D \sqcap E))}$$

where E is obtained by using Rule1 on premises C and D

Rule3 $$\frac{W_1 \sqcup (\leq nR.C) \qquad W_2 \sqcup (\geq kS.D)}{W_1 \sqcup W_2 \sqcup (\geq (k-n)S.dnf(D \sqcap \neg C))}$$

$n < k, S \sqsubseteq^* R$, $(\leq nR.C)$ is maximal in $W_1 \sqcup (\leq nR.C)$
and $(\geq kS.D)$ is maximal in $W_2 \sqcup (\geq kS.D)$

Rule4 $$\frac{W_1 \sqcup (\leq nR.C) \qquad W_2 \sqcup (\geq kS.D)}{W_1 \sqcup W_2 \sqcup (\leq (n-k)R.dnf(C \sqcap \neg D)) \sqcup (\geq 1S.dnf(D \sqcap \neg C))}$$

$n \geq k, S \sqsubseteq^* R$, $(\leq nR.C)$ is maximal in $W_1 \sqcup (\leq nR.C)$
and $(\geq kS.D)$ is maximal in $W_2 \sqcup (\geq kS.D)$

Figure 1: TBox inference rules of the DL calculus

S1 $$\frac{C \sqcup L \sqcup \cdots \sqcup L}{C \sqcup L}$$

S2 $$\frac{C \sqcup D \sqcup (D \sqcap E)}{C \sqcup D}$$

S3 $$\frac{C \sqcup D \sqcup (\neg D \sqcap E)}{C \sqcup D \sqcup E}$$

S4 $$\frac{C \sqcup D \sqcup \neg D}{\top}$$

S5 $$\frac{C \sqcup (D \sqcap E \sqcap \neg E)}{C}$$

S6 $$\frac{W \sqcup (\geq nR.\bot)}{W}$$

S7 $$\frac{W \sqcup (\leq nR.\bot)}{\top}$$

Figure 2: TBox simplification rules of the DL calculus

3.6 Saturation

We saturate the knowledge base, i.e., we apply the rules in Figure 1 to deduce new concepts as long as possible. Before adding the consequent to the concept set, we eagerly apply the simplification rules of Figure 2 to make the concept as simple as possible. We claim that the consequent is always a $\mathcal{SHQ}$-concept.

Proposition 2. *The set of $\mathcal{SHQ}$-concepts is closed under the inference rules in Figure 1 and the simplification rules in Figure 2.*

Proof. Consider Rule1. $D_1 \sqcap A$ is maximal in $C_1 \sqcup (D_1 \sqcap A)$ which is only possible if C_1 does not contain any $\geq$- or $\leq$-concepts. Hence it is a bool concept. Analogously, the fact that $D_2 \sqcap \neg A$ is maximal in $C_2 \sqcup (D_2 \sqcap \neg A)$ ensures that C_2 is another bool concept. Bool concepts are in DNF. The conclusion is the disjunction of two bool concepts $(C_1 \sqcup C_2)$ which is also in DNF and hence is a bool concept.

Rule2 resolves a bool concept with a $\geq$-max concept. We have just seen that resolving C and D by Rule1 yields a bool concept. We take the conjunction of this concept and another bool concept $(D \sqcap E)$ which is not in DNF, but it yields a bool concept once we apply the *dnf* operator. Hence the conclusion is a $\geq$-max concept.

In Rule3, the maximal disjunct of the first premise is $(\leq nR.C)$, so it does not contain any $\geq$-concept. The second premise is a $\geq$-max concept and contains exactly one $\geq$-concept, namely $(\geq kS.D)$. The conclusion contains one $\geq$-concept and is a $\geq$-max concept. Again, the *dnf* operator is used to ensure that the bool concept appearing in the $\geq$-disjunct of the conclusion is in DNF.

In Rule4, the maximal disjunct of the first premise is $(\leq nR.C)$, so it is a $\leq$-max concept and does not contain any $\geq$-concept. The second premise contains exactly one $\geq$-concept, so W_2 contains no $\geq$-concept. Consequently, the conclusion will contain only one $\geq$-concept and all subconcepts inside $\geq$- and $\leq$-concepts are bool concepts. We obtain a $\geq$-max concept.

Simplification rules S1-S5 eliminate some disjuncts or conjuncts from bool concepts in DNF. The conclusion is always a simpler bool concept in DNF. S6 eliminates an unsatisfiable branch from a disjunction, turning a $\geq$-max concept either to a bool concept or to a $\leq$-max concept. In case of S7, the premise is a tautology and can be safely eliminated. □

4 Termination

The following proposition – along with Propostion 2 – ensures that the DL calculus terminates.

Proposition 3. *The set of all $\mathcal{SHQ}$-concepts that can be deduced from any finite TBox is finite.*

Proof. For any finite TBox, there can only be finitely many distinct role expressions and bool concepts. Furthermore, note that each inference rule either leaves the arity of a number restriction unaltered or reduces it. So in a $(\leq nR.C)$ or $(\geq nR.C)$

expression the number of possible values for n, R and C is finite for a fixed TBox. As all $\mathcal{SHQ}$-concepts are disjunctions of bool, $\leq$, and $\geq$-concepts, we have an upper limit for the set of deducible $\mathcal{SHQ}$-concepts. $\square$

DL calculus deduces only $\mathcal{SHQ}$ concepts from $\mathcal{SHQ}$ concepts. Since there are finitely many $\mathcal{SHQ}$ concepts, even if we have to deduce every possible $\mathcal{SHQ}$-concept, it still requires finitely many steps, so the calculus is guaranteed to terminate.

5 Soundness

It is straightforward to show that the simplification rules are sound, i.e., if all individuals of an interpretation satisfy the premise then they also satisfy the conclusion. We leave this to the reader. The inference rules are slightly more complex.

Theorem 1. *The inference rules of the DL calculus are sound.*

Proof. Consider Rule1 and suppose that x satisfies both premises. Either A or $\neg A$ is true of x. If $A(x)$ is true, then x must satisfy C_2, due to the second premise. Analogously, if $\neg A(x)$ is true, then x must satisfy C_1. In either case, the conclusion holds for x.

We turn to Rule2. Let x be an individual. It satisfies the second premise, so either W or $(\geq nR.D)$ holds for x. In the first case the conclusion is satisfied by x, in the second case x has at least n R-successors that satisfy D. These successors also satisfy the first premise (C) and – given that Rule1 is sound – they satisfy E. If these R-successors satisfy both D and E, then they satisfy $D \sqcap E$ as well. So it holds for x that it has at least n R-successors that satisfy $D \sqcap E$, so the conclusion is again satisfied.

For Rule3, let x be an arbitrary indidivual. If x satisfies either W_1 or W_2, then it satisfies the conclusion. Otherwise, x satisfies $(\leq nR.C)$ and $(\geq kS.D)$, where $S \sqsubseteq R$. So, x has at least k distinct S-successors that satisfy D (that are R-successors as well). Of these, at most n successors can satisfy C, so there are at least $k - n$ S-successors that satisfy $\neg C$. From this it follows directly that the conclusion holds for x.

Finally, let us consider Rule4 and let again x denote an arbitrary individual. If x satisfies either W_1 or W_2, then it satisfies the conclusion. Otherwise, x satisfies $(\leq nR.C)$ and $(\geq kS.D)$, where $S \sqsubseteq R$. So, x has at least k distinct S-successors that satisfy D. If any of these successors satisfy $\neg C$ then the last disjunct of the conclusion holds. Otherwise, all the k S-successors satisfy C. Given that x can have no more than n successors that satisfy C, there cannot be more than $n - k$ successors that are not among those satisfying D, but they satisfy C. Hence the second to last disjunct of the conclusion holds for x. $\square$

6 The Completeness of the DL Calculus

In this section we prove that the method presented in Section 3 is complete, i.e., whenever there is some inconsistency in a TBox $\mathcal{T}$, the empty concept is deduced. We prove completeness by showing that if a saturated set $Sat_{\mathcal{T}}$ does not contain $\bot$ then the axiom $\top \sqsubseteq \bigsqcap Sat_{\mathcal{T}}$ has a model. Instead of building the model itself, we will prove that the $\mathcal{ALCHQ}$ tableau method can find one such model. In order for the model to satisfy $\top \sqsubseteq \bigsqcap Sat_{\mathcal{T}}$, the concepts in $Sat_{\mathcal{T}}$ are added to the label of every newly created node in the tableau.

Although the tableau rules are fairly standard, there might be small variations. Hence, to avoid confusion, in Appendix 8 we provide the definition of the tableau rules that we assume in the following.

6.1 Building the Tableau Tree

In the previous sections, we replaced $\forall$- and $\exists$-concepts with $\leq$- and $\geq$-concepts to make the presentation of the inference rules simpler. As we turn to the tableau, however, the reader might be more familiar with the corresponding $\forall$-rule and $\exists$-rule. Hence, in the following, we will treat our $(\leq 0R.C)$ and $(\geq 1S.D)$ concepts as $(\forall R.\neg C)$ and $(\exists S.D)$, respectively.

Whenever we have several applicable tableau rules, we require the following ordering precedence: $\sqcup$-rules, $\sqcap$-rule, $\exists$-rule, $\geq$-rule, $\forall$-rule, $\bowtie$-rule and $\leq$-rule. When applying the $\sqcup$-rule we proceed with the branch[1] that adds the minimal possible concept to the label of a node. Given that the tableau method is don't care non-deterministic with respect to these choices, the completeness of the algorithm is preserved.

Whenever a node n contains a disjunctive concept $W \sqcup C$, the branch where C is added to the label of n is only examined after each disjunct in W that is smaller than C has been proven unsatisfiable. A *clash* occurs in the tableau tree when an atomic concept name and its negation both appear in the label of some node. In this case we roll back and proceed with another branch. A *final clash* occurs when there are no branches left, i.e., the tableau proves the inconsistency of $Sat_{\mathcal{T}}$. We show that no final clash can be reached if $Sat_{\mathcal{T}}$ does not contain $\bot$.

6.2 Bool Concepts

In the following theorem we consider the case when $Sat_{\mathcal{T}}$ contains only bool concepts.

Theorem 2. *If $Sat_{\mathcal{T}}$ contains only bool concepts and does not contain $\bot$, then no final clash is possible.*

Proof. To obtain contradiction, suppose that we reach a final clash. Hence, for some atomic concept A, both A and $\neg A$ appear in the label of some node. This is

[1] Throughout this paper, "branch" refers to a branch of the meta-tableau tree, i.e., one of the tableaux resulting from the application of a non-deterministic rule.

only possible if $Sat_{\mathcal{T}}$ contains concepts

$$W_1 = C_1 \sqcup (D_1 \sqcap A) \qquad\qquad W_2 = C_2 \sqcup (D_2 \sqcap \neg A)$$

The clash is final, so there are no more branches, i.e., $(D_1 \sqcap A)$ and $(D_2 \sqcap \neg A)$ are maximal in W_1 and W_2, respectively, and each disjunct in C_1 and C_2 leads to clash. W_1 and W_2 are resolvable using Rule1, so $Sat_{\mathcal{T}}$ also contains

$$W = C_1 \sqcup C_2$$

W cannot be empty because we assumed that $Sat_{\mathcal{T}}$ does not contain $\bot$. The simplification rules, and in particular S1 was eagerly applied on W_1 and W_2, so there are no other occurrences of $(D_1 \sqcap A)$ in C_1 and $(D_2 \sqcap \neg A)$ in C_2. So the maximal disjuncts in W_1 and W_2 are strictly maximal. Let X denote the greater concept of $(D_1 \sqcap A)$ and $(D_2 \sqcap \neg A)$. X is greater than any disjunct in either C_1 or C_2. This means that the branches corresponding to all disjuncts of W were examined before examining the branch corresponding to X (due to the ordering imposed on the application of the $\sqcup$-rule described in Subsection 6.1). But we know that all disjuncts in W lead to clash, so a final clash must have been obtained on W, even before introducing X to the label of the node, which contradicts our assumption that the final clash involved X. $\square$

Corollary 1. *If $Sat_{\mathcal{T}}$ does not contain $\bot$, then the set of bool concepts in $\mathcal{T}$ is satisfiable.*

Notice that only Rule1 is used to detect the inconsistency of bool concepts. This observation will be useful for us later.

Corollary 2. *If a set N of bool concepts is unsatisfiable then there is a sequence of bool concepts $p_1, p_2 \dots p_n = \bot$ such that for each p_i, there is an instance of Rule1 with premises from $N \cup \{p_1, p_2 \dots p_{i-1}\}$ whose conclusion is p_i. We call this sequence a deduction of $\bot$.*

6.3 $\geq$-max Concepts

Let us now assume that $Sat_{\mathcal{T}}$ contains only bool concepts and $\geq$-max concepts.

Proposition 4. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept in $Sat_{\mathcal{T}}$. Then D is satisfiable.*

Proof. Suppose that D is unsatisfiable. Since it is in DNF, it is the disjunction of conjunctions such that each conjunction contains some atom together with its negation. However, the simplification rules are eagerly appled on all $\mathcal{SHQ}$-concepts and due to S5 all disjuncts of D were eliminated. Hence $D = \bot$ and $W = C \sqcup (\geq nR.\bot)$. S6 is applicable on W yielding C, so W was removed from $Sat_{\mathcal{T}}$ and replaced by C. This is a contradiction, so D must be satisfiable. $\square$

Proposition 5. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept and $B = \{B_i\}$ a set of bool concepts. If $\{D\} \cup B$ is inconsistent, then there is a deduction of C using Rule1 and Rule2 and the simplification rules.*

Proof. We know from Corollary 2 that there is a deduction $p_1, p_2 \dots p_n = \bot$ from $\{D\} \cup B$ using Rule1. In this sequence each concept has a set of premises, either from the original concept set or from concepts that were deduced earlier. Let us define the *ancestor* relation as the transitive closure of the premise relation and let *descendant* be its inverse relation. For each p_i, let A_i denote the set of its ancestors that are either identical to D or are descendants of D. For each p_i such that A_i is not the empty set, replace p_i with $C \sqcup (\geq nR.(p_i \sqcap \bigsqcap A_i))$. We obtain a deduction in which each time the conclusion is a $\geq$-max concept, Rule2 is used instead of Rule1. In particular, $p_n = \bot$ is replaced with $C \sqcup (\geq nR.(\bot \sqcap \bigsqcap A_n))$, where the $\geq$-concept is unsatisfiable, so we can deduce C from this concept using the simplification rules (see Proposition 4). □

Corollary 3. *Let $W = C \sqcup (\geq nR.D)$ be a $\geq$-max concept in $Sat_{\mathcal{T}}$ and let $B = \{B_i\}$ be the set of bool concepts in $Sat_{\mathcal{T}}$. Then $\{D\} \cup B$ is consistent.*

Proof. Suppose $\{D\} \cup B$ is inconsistent. Then, from Proposition 5, $Sat_{\mathcal{T}}$ contains C. However, C makes W redundant, so W was eliminated from $Sat_{\mathcal{T}}$ when C was added to it. This contradicts our assumption that $W \in Sat_{\mathcal{T}}$. □

Theorem 3. *If $Sat_{\mathcal{T}}$ contains only bool concepts and $\geq$-max concepts and does not contain $\bot$, then it is consistent.*

Proof. We know from Corollary 1 that the bool concepts are satisfiable. As of the $\geq$-max concepts, at least one of their disjuncts, namely the $\geq$-disjunct can be satisfied: we can create separate successors for each $\geq$-concept, independent of each other (without $\leq$-concepts, these successors never need to be identified). The label of each successor is satisfiable (see Proposition 4 and Proposition 3), so the $\geq$-concept in the parent is satisfiable as well. □

6.4 $\leq$-max Concepts

We now consider a fully general saturated set $Sat_{\mathcal{T}}$, that might contain bool concepts, $\geq$-max concepts and $\leq$-max concepts. When we build the tableau tree, if a $\leq$-concept appears in the label of a node, we possibly have to add a new concept to the label of a node ($\forall$-rule) or identify two nodes ($\leq$-rule). We show that none of these rules will lead to final clash.

Each successor node is created with an intial concept in its label: for instance, if a new node is created due to concept $\geq 1R.A$, then we call A the *creator concept* of the node. Whatever other concept appears in its label (before performing any identification step), it follows from $A \sqcap \bigsqcap B_i$, where $\{B_i\}$ is the set of bool concepts. If a node with creator concept A has to be identified with another such that the second node contains A in its label, then identification cannot introduce new inconsistency and it can be seen as simply deleting the first node.

As previously, we are only interested in potential clashes that are final. This means that the (non-disjunctive) concepts that are involved in the clash can be assumed to be the maximal disjuncts of $\mathcal{SHQ}$-concepts from $Sat_{\mathcal{T}}$.

Proposition 6. *Let $Sat_{\mathcal{T}}$ be a saturated set of $\mathcal{SHQ}$-concepts that does not contain the empty concept $\bot$. Let us try to build a model for $\top \sqsubseteq \bigsqcap Sat_{\mathcal{T}}$ using the tableau method, observing the restrictions on the order of rules presented in Subsection 6.1. Then we never obtain a final clash.*

Proof. We know from Theorem 3 that the set of bool concepts and $\geq$-max concepts is consistent. Hence, a final clash must involve a $(\leq nR.D)$ concept. We use induction on n, the arity of the $\leq$-concept to show that no final clash is possible.

The base case is when $n = 0$, that is, when we have a $\forall$-concept in the label of a node. The $\forall$-rule fires and a new concept is added to the label of some successors. To obtain contradiction, we assume that this leads to a final clash. Given a node x that has an S-successor y with creator concept A. This means that the label of x contains a concept $\geq kS.A$. Furthermore, the label of x also contains a $\forall$-concept, which is a $(\leq 0R.D)$ concept in our teminology. $S \sqsubseteq R$, so the $\forall$-rule is applicable and puts $\neg D$ in the label of y. We assumed that a clash is obtained, so $A \sqcap \neg D$ is not satisfiable. The $\geq$-concept and $\leq$-concept in the label of x originate from a $\geq$-max and a $\leq$-max concept, respectively, in $Sat_{\mathcal{T}}$, that is, $Sat_{\mathcal{T}}$ contains concepts

$$W = E \sqcup (\leq 0R.D) \qquad\qquad V = F \sqcup (\geq kS.A)$$

where $(\leq 0R.D)$ is maximal in W, $(\geq kS.A)$ is maximal in V and each disjunct in E and F leads to clash. W and V are resolvable using Rule3 and the conclusion is

$$E \sqcup F \sqcup (\geq kS.dnf(A \sqcap \neg D))$$

$A \sqcap \neg D$ is not satisfiable, so the DL calculus deduces $E \sqcup F$ as well (Proposition 5). However, we know that all disjuncts in E and F lead to clash, so we obtain a final clash without the $\leq$-concept in W. Contradiction.

We now turn to the inductive step. The inductive hypothesis is that a $\leq$-concept can never lead to final clash, i.e., a $(\leq n'R.D)$ concept in the label of a node that is derived from the maximal disjunct of a $\leq$-max concept of$Sat_{\mathcal{T}}$ can be satisfied for all $n' < n$. We show that this also holds for n.

Let some node x in the tableau tree contain concepts $(\leq nR.D)$ and $(\geq n_iS_i.A_i)$, where $1 \leq i \leq l$ and $S_i \sqsubseteq^* R$. Due to the $(\geq n_iS_i.A_i)$ concepts, we have already created $\Sigma_{i=1}^{l} n_i$ successors with creator concepts $A_1 \dots A_l$, respectively. D appears in the label of each S_i-successor, so A_i, together with the bool concepts implies D. This means that $A_i \sqcap \neg D$ is unsatisfiable. Suppose that we have to perform identification which leads to final clash. $Sat_{\mathcal{T}}$ contains concepts

$$W = E \sqcup (\leq nR.D) \qquad\qquad W_i = F_i \sqcup (\geq n_iS_i.A_i) \quad 1 \leq i \leq l$$

where $(\leq nR.D)$ is maximal in W, $(\geq n_iS_i.A_i)$ is maximal in W_i and each disjunct of E and F_i leads to clash in x. By the time a $\leq$-rule is applied, we have already performed all possible $\bowtie$-rules, due to which the label of each S_i-successor contains either A_j or $\neg A_j$ for all $j \in \{1 \dots l\}$. According to Corollary 3, each creator concept is satisfiable and hence will remain satisfiable by taking its conjunction with either A_i or $\neg A_i$.

We use induction on l, the number of $\geq$-concepts to show that the assumption that the $\leq$-concept gives rise to final clash leads to contradiction.

The base case (of the second, inner induction) is when $l = 0$. There are no $\geq$-concepts in the label of x, so there are no involved successors to be identified.

We now turn to the inductive step (of the inner induction). We assume that if the label of x contains only $l' < l$ different $\geq$-concepts then the resulting successors can be identified into n nodes without clash.

1. In case $n_l > n$ then Rule3 is applicable on W and W_l, resulting in:

   $$E \sqcup F_l \sqcup (\geq (n_l - n) S_l.dnf(A_l \sqcap \neg D))$$

   We know that $A_l \sqcap \neg D$ is unsatisfiable, so the DL calculus deduces $E \sqcup F_i$ (from Proposition 5). However, all disjuncts of E and F_i lead to clash in x, so we obtain a final clash even before introducing any $\leq$- and $\geq$-concept, contrary to our assumption.

2. If $n \geq n_l$, then concepts W and W_l are resolvable using Rule4, resulting in

   $$E \sqcup F_l \sqcup (\leq (n - n_l) R.dnf(D \sqcap \neg A_l)) \sqcup (\geq 1 S_l.dnf(A_l \sqcap \neg D))$$

   Again, we know that $D \sqcap \neg A_l$ is unsatisfiable, so (from Proposition 5) the DL calculus deduces

   $$E \sqcup F_l \sqcup (\leq (n - n_l) R.dnf(D \sqcap \neg A_l)) \qquad (1)$$

   Due to the $\bowtie$-rule, the label of every successor contains either A_l or $\neg A_l$. $n - n_l < n$, so the inductive hypothesis holds for (1), i.e., all the successors whose label contains both D and $\neg A_l$ can be identified into $n - n_l$ nodes by deleting some successors that are not necessary. Further to this, there are n_l successors with creator concept A_l, plus some k other successors such that the $\bowtie$-rule put A_l into their labels.

   a) If $k \leq n_l$ then we can eliminate $n_l - k$ nodes from those having A_l as their creator concept, leaving exactly n_l successors whose label contains A_l. Contrary to our assumption, we obtain no final clash.

   b) If $k > n_l$ then each of the nodes whose creator concept is A_l can be eliminated since there are more then n_l other nodes satisfying A_l. All remaining successors originate from the $\geq$-concepts in $W_1 \dots W_{l-1}$. However, according to the inductive hypothesis (of the inner induction), these successors can be identified into n successors without clash.

This concludes the second inductive proof and the first one as well. We have showed that the assumption that a $\leq$-concept introduces inconsistency into the label of a node leads to contradiction. $\square$

6.5 Conclusion

Let $\mathcal{T}$ be a $\mathcal{SHQ}$ TBox. Let $Sat_{\mathcal{T}}$ be the set of concepts obtained after performing preprocessing on $\mathcal{T}$ and then saturating it with the DL calculus. In the preceding subsections we have showed that if $Sat_{\mathcal{T}}$ does not contain $\perp$ then it is possible to build a model for $\mathcal{T}$ using the tableau algorithm. This concludes the proof of completeness for the DL calculus.

7 Towards a DL Calculus for ABox Reasoning

In this section we sketch an extension to the DL calculus that performs ABox reasoning. Although the answers that we obtained in our test queries are identical to those of other reasoners, we do not yet have a formal proof of completeness. Accordingly, this section should be seen as an indication of our future work. Currently, the DLog data reasoner uses the calculus described in [12] and the DL calculus is only in test phase.

We restrict our attention to *extensionally reduced* ABoxes, i.e., we assume that the concept assertions only contain literal concepts. An arbitrary knowledge base can be easily transformed to satisfy this constraint. Furthermore, we use the Unique Name Assumption (UNA): we assume that different names refer to different individuals. The rules in Figure 3 are added to the inference rules presented in Figure 1. As before, the resolved literals have to be maximal in their respective concepts.

$$\mathbf{5}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L) \qquad \neg L(a)}{C(a)} \qquad \mathbf{6}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L(a)) \qquad \neg L(a)}{C}$$

$$\mathbf{7}\ \frac{W \sqcup (\leq nR.C) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W(a) \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)} \qquad \mathbf{8}\ \frac{W \sqcup (\leq nR.C)(a) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)}$$

Figure 3: ABox inference rules

An important property of the ABox DL calculus is that the rules for data axioms do not involve $\geq$-max concepts. This suggests that all inference steps involving $\geq$-max concepts can be performed *before accessing the ABox*, allowing us to break the ABox reasoning into two parts: (1) an ABox independent DL calculus is first applied to the TBox until all the consequences of $\geq$-max concepts are inferred; (2) next we perform the actual data reasoning using a much simpler TBox (as all $\geq$-max concepts can be eliminated). The output of the first phase is translated to first-order clauses during data reasoning and $\geq$-concepts give rise to skolem functions. Without $\geq$-max concepts, we can work with function-free clauses, which makes the reasoning task much easier [8].

8 Conclusion and Future Work

We have presented the DL calculus, a resolution based algorithm for deciding the consistency of a $\mathcal{SHQ}$ TBox. The novelty of this calculus is that it is defined directly on DL axioms. We showed that the algorithm is sound, complete and terminates. More work needs to be done to explore the real time complexity of the reasoning, as well as potential optimization techniques. We hope that further research will reveal that the DL calculus provides a reasonable alternative to the Tableau Method for certain reasoning tasks.

We have extended the DL calculus to consider ABox axioms as well, providing the basis of a two-phase ABox reasoning framework. The DL calculus is used to perform the first phase involving the TBox only. Given that the TBox is relatively stable over time, the speed of this phase is not crucial as it has to be performed only once. What really matters is that by the end of this phase we can eliminate many axioms that make the knowledge base much simpler. Thanks to the eliminations, we can translate the initial knowledge base into a set of first-order clauses that are function-free. The absence of function symbols enables us to use the query driven, highly efficient data reasoning techniques implemented in the DLog ABox reasoner. It has to be noted, however, that the ABox extension of the DL calculus requires further work.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2004.

[2] Bachmair, L. and Ganzinger, H. Resolution theorem proving. In Robinson, A. and Voronkov, A., editors, *Handbook of Automated Reasoning*, volume 1, chapter 2, pages 19–100. North Holland, 2001.

[3] Colmerauer, Alain and Roussel, Philippe. The birth of prolog. In Bergin, Jr., Thomas J. and Gibson, Jr., Richard G., editors, *History of programming languages—II*, pages 331–367. ACM, New York, NY, USA, 1996.

[4] Haarslev, V. and Möller, R. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5*, pages 163–173, 2004.

[5] Haarslev, V., Möller, R., van der Straeten, R., and Wessel, M. Extended Query Facilities for Racer and an Application to Software-Engineering Problems. In *Proceedings of the 2004 International Workshop on Description Logics (DL-2004), Whistler, BC, Canada, June 6-8*, pages 148–157, 2004.

[6] Horrocks, Ian, Sattler, Ulrike, and Tobies, Stephan. A description logic with transitive and converse roles, role hierarchies and qualifying number restrictions. LTCS-Report 99-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999.

[7] Hustadt, Ullrich, Motik, Boris, and Sattler, Ulrike. Reasoning for Description Logics around SHIQ in a resolution framework. Technical report, FZI, Karlsruhe, 2004.

[8] Lukácsy, Gergely and Szeredi, Péter. Efficient description logic reasoning in Prolog: the DLog system. *Theory and Practice of Logic Programming*, 09(03):343–414, May 2009.

[9] Lukácsy, Gergely, Szeredi, Péter, and Kádár, Balázs. Prolog based description logic reasoning. In de la Banda, Maria Garcia and Pontelli, Enrico, editors, *Proceedings of 24th International Conference on Logic Programming (ICLP'08),Udine, Italy*, pages 455–469, December 2008.

[10] Motik, Boris. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany, January 2006.

[11] Sirin, Evren, Parsia, Bijan, Grau, Bernardo Cuenca, Kalyanpur, Aditya, and Katz, Yarden. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[12] Zombori, Zsolt. Efficient two-phase data reasoning for description logics. In Bramer, Max, editor, *IFIP AI*, volume 276 of *IFIP*, pages 393–402. Springer, 2008.

[13] Zombori, Zsolt and Lukácsy, Gergely. A resolution based description logic calculus. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik and Sattler, Ulrike, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford*, volume 477 of *CEUR Workshop Proceedings*, Oxford, UK, July 27-30 2009. `http://CEUR-WS.org/Vol-477` (accessed May 18, 2010).

Appendix: ALCHQ tableau rules

In this appendix we provide the rules of the tableau method. Even though the TBox reasoning starts out from an $\mathcal{SHQ}$ knowledge base, we quickly eliminate transitivity axioms during preprocessing and obtain an $\mathcal{ALCHQ}$ knowledge base. Accordingly, the rules provided in Figures 4 and 5 are those for the $\mathcal{ALCHQ}$ language. This appendix is not meant to explain how the tableau works. Instead, we provide it to make explicit what sorts of tableau rules we assume. For a comprehensive treatment of $\mathcal{SHIQ}$-tableau, we refer the reader to [6].

$\sqcap$-rule	
Condition:	$(C_1 \sqcap C_2) \in \mathcal{L}(x)$, x is not indirectly blocked and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$.
New state $\mathbf{T}'$:	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$.
$\sqcup$-rule	
Condition:	$(C_1 \sqcup C_2) \in \mathcal{L}(x)$, x is not indirectly blocked and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$.
New state $\mathbf{T}_1$:	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_1\}$.
New state $\mathbf{T}_2$:	$\mathcal{L}'(x) = \mathcal{L}(x) \cup \{C_2\}$.
$\exists$-rule	
Condition:	$(\exists R.C) \in \mathcal{L}(x)$, x is not blocked and x has no R-neighbour y for which $C \in \mathcal{L}(y)$.
New state $\mathbf{T}'$:	$V' = V \cup \{y\}$ ($y \notin V$ is a new node), $E' = E \cup \{(x, y,)\}$, $\mathcal{L}'((x, y,)) = \{R\}$, $\mathcal{L}'(y) = \{C\}$.
$\forall$-rule	
Condition:	$(\forall R.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and x has an R-neighbour y for which $C \notin \mathcal{L}(y)$.
New state $\mathbf{T}'$:	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{C\}$.

Figure 4: The transformation rules of the $\mathcal{ALCHQ}$ tableau algorithm, part 1.

$\bowtie$- rule	
Condition:	$(\bowtie nR.C) \in \mathcal{L}(x)$, where $\bowtie$ is one of the symbols $\geq$ or $\leq$, x is not indirectly blocked, and x has an R-neighbour y for which $\{C, \sim C\} \cap \mathcal{L}(y) = \emptyset$.
New state* $\mathbf{T}_1$*:	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{C\}$.
New state* $\mathbf{T}_2$*:	$\mathcal{L}'(y) = \mathcal{L}(y) \cup \{\sim C\}$.
$\geq$-rule	
Condition:	$(\geq n\, R.C) \in \mathcal{L}(x)$, x is not blocked, and it is not the case that there exist nodes $y_1, \ldots, y_n$ such that no two of them are identifiable, and for every i, y_i is an R-neighbour of x, and $C \in \mathcal{L}(y_i)$ holds.
New state* $\mathbf{T}'$*:	$V' = V \cup \{y_1, \ldots, y_n\}$ ($y_i \notin V$ new nodes), $E' = E \cup \{(x, y_1, ,) \ldots, (x, y_n, \})$, $\mathcal{L}'((x, y_i,)) = \{R\}$, $\mathcal{L}'(y_i) = \{C\}$, for every $i = 1 \leq i \leq n$, $I' = I \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n\}$.
$\leq$-rule	
Condition:	$(\leq n\, R.C) \in \mathcal{L}(x)$, x is not indirectly blocked, x has $n+1$ R-neighbours $y_0, \ldots, y_n$ such that $C \in \mathcal{L}(y_i)$ holds for every i, and there exist y_i and y_j that are identifiable.
For every $(0 \leq i < j \leq n)$, where y_i and y_j are identifiable, let $\{y, z\} = \{y_i, y_j\}$ so that x is not a successor of y:	
New state* $\mathbf{T}_{ij}$*:	$\mathcal{L}'(z) = \mathcal{L}(z) \cup \mathcal{L}(y)$, $\mathcal{L}'((x, y,)) = \emptyset$, $\mathcal{L}'((z, x,)) = \mathcal{L}((z, x,)) \cup \mathsf{Inv}(\mathcal{L}((x, y,)))$ if x is a successor of z, $\mathcal{L}'((x, z,)) = \mathcal{L}((x, z,)) \cup \mathcal{L}((x, y,))$ if x is not a successor of z, $I' = I[y \to z]$ (each occurrence of y is replaced by z).

Figure 5: The transformation rules of the $\mathcal{ALCHQ}$ tableau algorithm, part 2.