A Resolution Based Description Logic Calculus

Zsolt Zombori and Gergely Lukácsy

Budapest University of Technology and Economics
Department of Computer Science and Information Theory
1117 Budapest, Magyar tudósok körútja 2., Hungary
{zombori,lukacsy}@cs.bme.hu

Abstract. We present a resolution based reasoning algorithm called *DL calculus* that decides concept satisfiability for the $\mathcal{SHIQ}$ 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 $\mathcal{SHIQ}$ tableau method, without requiring any further background knowledge.

Besides pure terminology reasoning, the DL calculus can be extended to ABox reasoning, providing the basis for an efficient two-phase data reasoning framework. Here, the DL calculus is used in the first phase to simplify the content of the TBox by (1) performing those reasoning steps that can be done independently of the ABox and (2) eliminating the axioms that cannot participate in the ABox reasoning. The content of the ABox is only accessed in the second phase when the simplified TBox allows for answering instance retrieval queries in a focused, efficient way. The above mentioned techniques are implemented in the $\mathcal{SHIQ}$ ABox reasoner called DLog, available to download from SourceForge.

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. [2]), that are used by the state-of-the-art DL reasoners, such as RacerPro [3] or Pellet [4].

Other approaches use first-order resolution for reasoning. A resolution-based inference algorithm is described in [5] which is not as sensitive to the increase of the ABox size as the tableau-based methods. The system KAON2 [6] 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 to 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 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 [7], 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 [8] to perform the second phase (see [9]). Second, we use a modified resolution calculus (see [10]) that allows us to perform more inference steps in the first phase, thanks to which more axioms can be eliminated, yielding to an even simpler set of axioms to work with in the second phase. The important difference is that while the approach of [6] 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.

Our current work with the DL calculus aims to improve on the calculus presented in [10]. We move the resolution-based reasoning from the level of first-order clauses to DL axioms, which saves us many intermediary transformation steps, while preserving the main merit of the calculus, namely that we can eliminate a large part of the TBox before accessing the ABox.

This paper is structured as follows. In Section 2 we present the DL calculus that performs consistency check for a $\mathcal{SHIQ}$ TBox, and show that it can also be used to decide concept satisfiability. In Section 3 we prove that the calculus is complete. Section 4 extends the DL calculus to ABox reasoning. Finally, Section 5 concludes by giving a brief summary of our results.

2 The DL Calculus

In this section we present a reasoning algorithm called *DL calculus* which decides the consistency of a $\mathcal{SHIQ}$ TBox. Before doing so, however, we show that such an algorithm can be used for deciding concept satisfiability as well.

Suppose we want to know whether concept C is satisfiable in the presence of TBox T. Let R be a role that appears neither in T nor in C. Let us consider a new TBox $T' = T \cup \{\top \sqsubseteq \exists R.C\}$. 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 T if and only if 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 one or 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 [11]. 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 ($\bot$).

2.1 Example derivation

To illustrate our calculus, we show how to derive the empty concept from an inconsistent $\mathcal{SHIQ}$ TBox $\mathcal{T}$. Suppose that $\mathcal{T}$ forces the following concepts to be true on all individuals (i.e. $\mathcal{T}$ contains the axioms $\top \sqsubseteq C_i, 1 \leq i \leq 6$):

$$
\begin{aligned}
C_1 &: \quad \exists hasChild.Clever \\
C_2 &: \quad \exists hasChild^-.Rich \\
C_3 &: \quad \leq 1 hasChild.Tall \\
C_4 &: \quad \leq 1 hasChild.Fat \\
C_5 &: \quad Fat \sqcup Tall \\
C_6 &: \quad \exists hasFriend.(\neg Clever \sqcap Fat \sqcap Tall)
\end{aligned}
$$

The derivation proceeds through the following steps. To anticipate the DL calculus, we give the corresponding inference rules, together with the premise concepts (for compactness we abbreviate $hasChild$ with hc):

$$
\begin{aligned}
C_7 &: \quad \forall hc.(Clever \sqcup \neg Tall) \sqcup \exists hc.(Clever \sqcap \neg Tall) && (\text{Rule4: } C_1, C_3) \\
C_8 &: \quad \forall hc.(Clever \sqcup \neg Tall) \sqcup \exists hc.(Clever \sqcap Fat) && (\text{Rule2: } C_7, C_5) \\
C_9 &: \quad \forall hc.(Clever \sqcup \neg Tall) \sqcup \forall hc.(Clever \sqcup \neg Fat) && (\text{Rule4: } C_8, C_4) \\
C_{10} &: \quad \forall hc.(Clever \sqcup \neg Fat \sqcup \neg Tall) && (\text{Rule6: } C_9) \\
C_{11} &: \quad Clever \sqcup \neg Fat \sqcup \neg Tall && (\text{Rule5: } C_{10}, C_2) \\
C_{12} &: \quad \exists hasFriend.\bot && (\text{Rule2: } C_{11}, C_6) \\
C_{13} &: \quad \bot && (C_{12})
\end{aligned}
$$

If someone has a clever child and has at most one tall child, then either he does not have any tall and not clever children, or else he also has a clever and not

tall child ($C_1 + C_3 \to C_7$). If he has a clever and not tall child, then this child is clever and fat ($C_7 + C_5 \to C_8$). Given that nobody can have more than one fat child, the person having a clever and fat child can have no fat and not clever children ($C_8 + C_4 \to C_9$). We obtain that for any individual, either he does not have any tall and not clever children, or he does not have any fat and not clever children. In particular, this entails that he cannot have any tall and fat and not clever children ($C_9 \to C_{10}$). We know that every individual has a rich parent and it is true for the parent that he cannot have a child that is tall and fat and not clever, so the individual itself cannot be tall and fat and not clever, so he is either not tall or not fat or clever ($C_{10} + C_2 \to C_{11}$). Hence, nobody can have a child that is tall and fat and not clever ($C_{11} + C_6 \to C_{12}$), so we obtained a contradiction ($C_{12} \to C_{13}$).

2.2 Preprocessing

We first eliminate transitivity from the knowledge base as described in [6]. Next, we internalize the TBox, i.e., we transform all axioms into a set of concepts that have to be satisfied by each individual. This is followed by structural transformation which eliminates the nesting of composite concepts into each other [6]. Finally, we make a small syntactic transformation: concepts $\forall R.C$ and $\exists R.D$ are replaced by their 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 expression:

$$L_1 \sqcup L_2 \sqcup \cdots \sqcup L_i$$
$$L_1 \sqcup (\geq kR.L_2)$$
$$L_1 \sqcup (\leq nR.L_2)$$

A derivation in the DL calculus generates concepts that are slightly more general. We call this broader set $\mathcal{SHIQ}$-*concepts*, defined as follows (C, D, E stand for concepts containing no role expressions):

$$C$$
$$C \sqcup \bigsqcup (\leq nR.D)$$
$$C \sqcup (\geq kS.E) \sqcup \bigsqcup (\leq nR.D)$$

2.3 Notation

Before presenting the inference rules, we define some important notions. A *literal concept* is a possibly negated atomic concept. A *bool concept* contains no role expressions (only negation, union and intersection are allowed). 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 $\bot$. We also impose a partial ordering ($\succ$) on DL concept expressions, as follows:

Definition 1 (concept ordering).

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. *$C_1 = (\leq n_1 R_1.A_1)$ is greater than $C_2 = (\leq n_2 R_2.A_2)$ 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 $A_1 \succ A_2$*
4. *For atomic roles, $R \succ S$ if R is lexicographically greater than S.*
5. *For atomic concepts, $A \succ B$ if A is lexicographically greater than B.*
6. *$\neg A \succ A$*
7. *$\neg A \succ B$ if $A \succ B$*
8. *$A \succ \neg B$ if $A \succ B$*
9. *$\bigsqcup A_i \succ \bigsqcup B_i$ if $\exists i(\forall j(A_i \neq B_j \wedge (B_j \succ A_i \rightarrow \exists l(A_l = B_j))))$*
10. *$\bigsqcap A_i \succ \bigsqcap B_i$ if $\exists i(\forall j(A_i \neq B_j \wedge (B_j \succ A_i \rightarrow \exists l(A_l = B_j))))$*

In the above we allow $\bigsqcup A_i$ ($\bigsqcap A_i$) to be singular, i.e., it can stand for a non-disjunctive (non-conjunctive) concept.

Definition 2 (maximal concept). *In a set N of concepts, concept $C \in N$ is maximal if it is greater than any other concept in N.*

The $\succ$-ordering given in Definition 1 is not total. However, any two disjuncts of a $\mathcal{SHIQ}$-concept (cf. Section 2.2) are comparable, hence, for any disjunctive $\mathcal{SHIQ}$-concept we can talk about the maximal disjunct.

2.4 Inference Rules

The inference rules are presented in Figure 1, where L_i stand for literal concepts, C, D, E are bool concepts and W is a $\mathcal{SHIQ}$-concept without $\geq$-concept. The rules manipulate on one disjunct of each premise and we make the further restriction that the involved disjunct must be maximal. Some of the rules do not preserve the DNF form of bool concepts. In such cases, an additional step is required to transform the bool concepts appearing in the consequent into DNF form. For the sake of simplicity, these additional steps are omitted from Figure 1.

It is straightforward to show that the inference rules are sound, i.e., that any interpretation satisfying the premises will necessarily satisfy the conclusion as well. We leave this to the reader.

2.5 Saturation

We saturate the knowledge base, i.e., we apply the rules in Figure 1 to deduce new concepts as long as possible. We claim that saturation terminates. For any TBox with finite signature, 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 bound for a fixed TBox. As all $\mathcal{SHIQ}$-concepts are disjunctions of bool, $\leq$, and $\geq$-concepts, we have an upper limit for the set of deducible $\mathcal{SHIQ}$-concepts: generating every possible $\mathcal{SHIQ}$-concept still requires only finitely many steps.

$$\textbf{Rule1} \quad \cfrac{C \sqcup (L_{C_1} \sqcap L_{C_2} \cdots \sqcap L) \qquad D \sqcup (L_{D_1} \sqcap L_{D_2} \cdots \sqcap \neg L)}{C \sqcup D}$$

$$\textbf{Rule2} \quad \cfrac{W \sqcup (\geq nR.C) \qquad D}{W \sqcup (\geq nR.(C \sqcap E))}$$

where E is obtained by resolving C and D using Rule1

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

$n < k, S \sqsubseteq^* R$

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

$n \geq k, S \sqsubseteq^* R$

$$\textbf{Rule5} \quad \cfrac{D \sqcup (\leq 0R.C) \qquad W \sqcup (\geq kS.E)}{W \sqcup \neg C \sqcup (\geq kS.(E \sqcap D))}$$

where $\text{inv}(S) \sqsubseteq^* R$ and D is a bool concept

$$\textbf{Rule6} \quad \cfrac{W \sqcup (\leq 0R.C) \sqcup (\leq 0R.D)}{W \sqcup (\leq 0R.(C \sqcap D))}$$

Fig. 1. TBox inference rules of the DL calculus

3   The Completeness of the DL Calculus

In this section we prove that the method presented in Section 2 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 $\mathcal{SHIQ}$ tableau method can find one such model; in the course of the tableau method the concepts of $Sat_{\mathcal{T}}$ are added to the label of every newly created node.

3.1   Building the Tableau Tree

We require the following ordering on the tableau rules: $\sqcup$-rule, $\sqcap$-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.

The concepts of $Sat_{\mathcal{T}}$ give rise to the appearance of literal concepts in labels of nodes. Depending on what type of $\mathcal{SHIQ}$-concept was involved, we say that a literal concept was *derived* from a bool concept, a $\leq$-concept or a $\geq$-concept.

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 proved unsatisfiable. A *clash* occurs in the tableau tree

---

[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.

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 $\perp$.

Let us assume that the tableau method reaches a final clash. Literal concepts X and $\neg X$ appear in the label of a node and there are no other branches to be examined. We examine in turn the different concepts from which the clash could have been derived and show contradiction.

3.2 Bool Concepts

Suppose that X and $\neg X$ were derived from bool concepts, i.e., $Sat_{\mathcal{T}}$ contains concepts

$$W_C = C_1 \sqcup (C_2 \sqcap X) \qquad\qquad W_D = D_1 \sqcup (D_2 \sqcap \neg X)$$

Since there are no more branches, $(C_2 \sqcap X)$ and $(D_2 \sqcap \neg X)$ are maximal in W_C and W_D, respectively, and each disjunct in C_1 and D_1 leads to a clash. W_C and W_D are resolvable using Rule1, so $Sat_{\mathcal{T}}$ contains

$$W = C_1 \sqcup D_1$$

W cannot be empty because we assumed that $Sat_{\mathcal{T}}$ does not contain $\perp$. Should W contain X or $\neg X$, we can resolve it with either W_C or W_D resulting in a concept W_2 that makes W redundant. This follows from conditions 6-10 of Definition 1 and the fact that X and $\neg X$ are maximal in W_C and W_D. After possibly several such inference steps we obtain a concept W_i that contains neither X nor $\neg X$. Let us assume that $W = W_i$, i.e., we derived from W_C and W_D a concept W that does not contain X or $\neg X$. Both X and $\neg X$ are greater than any disjunct in W. This means that before introducing X we have already examined all branches corresponding to W. We assumed that each of these branches leads to a clash, hence a final clash was reached before X appeared in the label. This is a contradiction.

3.3 The Application of the $\geq$-rule

Let us now assume that one of the contradicting concepts, say X was derived from a $\geq$-concept and $\neg X$ from a bool concept. $Sat_{\mathcal{T}}$ contains concepts

$$W_C = C_1 \sqcup (\geq nR.(C_2 \sqcup (C_3 \sqcap X))) \qquad\qquad W_D = D_1 \sqcup (D_2 \sqcap \neg X)$$

Since there are no more branches, $(\geq nR.(C_2 \sqcup (C_3 \sqcap X)))$ and $(D_2 \sqcap \neg X)$ are maximal in W_C and W_D, respectively, $(C_3 \sqcap X)$ is maximal in $(C_2 \sqcup (C_3 \sqcap X))$ and each disjunct in C_1, C_2 and D_1 leads to a clash. W_C and W_D are resolvable in the calculus using Rule2, so $Sat_{\mathcal{T}}$ contains

$$W = C_1 \sqcup (\geq nR.(C_2 \sqcup (D_1 \sqcap C_3 \sqcap X)))$$

W makes W_C redundant, so the only way the latter can be present in $Sat_{\mathcal{T}}$ is that $W_C = W$, i.e., D_1 is one of the conjuncts of C_3. We know that C_1, C_2 and D_1 all lead to clash, so we obtain a final clash on $W_C = W$ without W_D, contradicting our assumption.

3.4 The Application of the $\forall$-rule

We now examine the case when one of the concepts participating in the final clash is derived from a $(\leq)$-concept with zero arity, i.e., a $\forall$-concept. The $\forall$-rule adds a concept to the label of either a successor or the predecessor.

In the first case some successors have been created, so the node containing the $\forall$-concept also contains a $\geq$-concept, i.e., $Sat_{\mathcal{T}}$ contains concepts

$$W_C = C_1 \sqcup (\geq nS.C) \qquad W_D = D_1 \sqcup (\leq 0R.D)$$

where $S \sqsubseteq^* R$. As previously, we assume that $(\geq nS.C)$ is maximal in W_C, $(\leq 0R.D)$ is maximal in W_D and each disjunct in C_1 and D_1 leads to clash. W_C and W_D can be resolved using Rule3, resulting in

$$W = C_1 \sqcup D_1 \sqcup (\geq nS.(C \sqcap \neg D))$$

By the time the $\geq$-concept from W_C and the $\leq$-concept from W_D appear in the label of a node, the $\geq$-concept from W also appears in the label. Hence, we create another set of n S-successors (the $\geq$-rule is applied prior to the $\forall$-rule), with label $(C \sqcap \neg D)$. Should this be inconsistent, we obtain a clash before the $\forall$-rule. Consequently, adding $\neg D$ to the first set of n S-successors of the node will not introduce any clash that would not have occurred otherwise.

In the second case the $\forall$-rule fires upwards and a bool concept is added to the predecessor. This happens when a node contains concept $(\geq nS.D)$ and one of the thus created successors contains a concept $(\leq 0R.C)$ where $\text{inv}(S) \sqsubseteq^* R$. That is, $Sat_{\mathcal{T}}$ contains the following concepts:

$$W_C = C_1 \sqcup (\geq nS.C) \qquad W_D = D_1 \sqcup (\leq 0R.D)$$

$(\geq nR.C)$ is maximal in W_C, $(\leq 0S.D)$ is maximal in W_D, each disjunct in C_1 leads to clash in the parent and each disjunct in D_1 leads to clash in the successor. Either D_1 does not contain any $\leq$-concepts, or Rule6 is applicable to it yielding to a concept $W_{D'} = D_{1'} \sqcup (\leq 0R.D')$, where $D_{1'}$ does not contain any $\leq$-concepts. In the latter case, let $W_D = W_{D'}$. W_C and W_D can be resolved by Rule7, resulting in

$$W = C_1 \sqcup \neg D \sqcup (\geq nS.(C \sqcap D_1))$$

By the time the $\forall$-rule in the successor fires, the $\sqcup$-rule has already been applied on the parent, so one of the disjuncts of W appears in the label of the latter. It cannot be a disjunct of C_1 because they all lead to clash. If it is a disjunct of $\neg D$ then the $\forall$-rule in the successor does not fire. Finally, if the parent contains the concept $(\geq nS.(C \sqcap D_1))$, that must have lead to a final clash because we know that D_1 together with C leads to clash in a successor. In conclusion, by the time the $\forall$-rule would be applicable, it either does nothing or we have already obtained a final clash.

3.5 Application of the $\leq$-rule

It remains to be seen if we might obtain a final clash through a $\leq$-rule, i.e., the identification of some nodes. Note that by the time a $\leq$-rule is applied, we have already performed all possible $\bowtie$-rules. We will show that in case of an identification that could potentially lead to a final clash, the identification is not only possible, but it can be performed simply by eliminating some successors, without having to add any concepts to the label of the remaining nodes[2].

Let some node contain concepts $(\leq nR.D)$ and $(\geq n_iS_i.A_i)$, where $1 \leq i \leq l$ and $S_i \sqsubseteq^* R$. Suppose that $\Sigma_{i=1}^l n_i > n$, so we need to perform identification and failing to do so would lead to a final clash. This can only happen when $Sat_{\mathcal{T}}$ contains concepts

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

where $(\leq nR.D)$ is maximal in W, $(\geq n_iS_i.A_i)$ is maximal in W_i, each disjunct in E and F_i leads to clash and D can be deduced from each A_i (using the bool concepts in $Sat_{\mathcal{T}}$). We will also consider the case when the node has an R-predecessor, i.e., when it is the S_0-successor of some node and $\text{inv}(S_0) \sqsubseteq^* R$. By using induction on n, i.e., the arity of the $\leq$ concept, we show that identification is possible by simply deleting some nodes in the tableau.

The base case, when $n = 0$ has already been partly discussed in the section of the $\forall$-rule. We have seen that whenever the $\forall$-rule would have to put a concept C in the label of some successors, there are other successors having the same label plus C. This means that we can simply delete the successors whose label do not contain C. We also need to examine what happens when we have an R-predecessor. Then, the node was created due to a $(\geq kS_0.X)$ concept in the parent, where $\text{inv}(S_0) \sqsubseteq^* R$. So, $Sat_{\mathcal{T}}$ contains concepts

$$W = E \sqcup (\leq 0R.D) \qquad V = G \sqcup (\geq kS_0.X)$$

where E is false in the examined node and G is false in the parent. The above two concepts can be resolved using Rule5, resulting in

$$G \sqcup \neg D \sqcup (\geq kS_0.(X \sqcap E))$$

One disjunct of this concept is satisfied by the parent and it cannot be G. If the parent satisfies $\neg D$ then it is not affected by the $(\leq 0R.D)$ concept. If the concept $(\geq kS_0.(X \sqcap E))$ appears in its label, there will be successor nodes with label $X \sqcap E$. The examined node was created with concept X and we know that the branch where E was added to the label failed, so $X \sqcap E$ is not satisfiable. So, should the label of the parent really contain $(\geq kS_0.(X \sqcap E))$, we would have obtained a clash even before applying the $\leq$-rule, contrary to our assumption.

For the inductive step, assume that any $\leq$-concept with arity smaller than n can be satisfied by deleting some nodes. Concepts W and W_1 in $Sat_{\mathcal{T}}$ are

---

[2] We might, however, have to extend the label of an edge due to identification, but this cannot lead to clash.

resolvable using Rule4, resulting in

$$E \sqcup F_1 \sqcup (\leq (n - n_1)R.(D \sqcap \neg A_1)) \sqcup (\geq 1 S_1.(A_1 \sqcap \neg D))$$

D is deducible from A_1 using bool concepts of $Sat_{\mathcal{T}}$, so the calculus also deduces

$$E \sqcup F_1 \sqcup (\leq (n - n_1)R.(D \sqcap \neg A_1)) \tag{1}$$

Due to the $\bowtie$-rule, the label of every successor contains either A_1 or $\neg A_1$. $n - n_1 < n$, so the inductive hypothesis applies to (1), i.e., all the successors whose label contains both D and $\neg A_1$ can be identified into $n - n_1$ nodes without altering the labels of the remaining nodes. Further to this, there are n_1 successors created with the $\geq$-concept in W_1 – we say that A_1 is the *creator concept* of these nodes, – plus some k other successors such that the $\bowtie$-rule put A_1 into their labels.

1. If $k \leq n_1$ then we can eliminate $n_1 - k$ nodes from those having A_1 as their creator concept, leaving exactly n_1 successors whose label contains A_1.
2. If $k > n_1$ then each of the nodes whose creator concept is A_1 can be eliminated since there are more then n_1 other nodes satisfying A_1. In case we still have too many successors left, we repeat the above argument on the successors originating from concepts $W_2, W_3, \ldots W_l$. Again, either we find that eliminating some of the successors with creator concept A_i gives a good identification, or we can eliminate all the successors with creator concept A_i. In the latter case we proceed with A_{i+1} etc. until at most n successors remain.

In case $n_i > n$ then Rule3 is applied instead of Rule4 when resolving W and W_i, resulting in:

$$E \sqcup F_i \sqcup (\geq (n_i - n)S_i.(A_i \sqcap \neg D))$$

Since D is deducible from A_i, the calculus deduces $E \sqcup F_i$. We know that all disjuncts in E and F_i are false, so we obtained a final clash even before introducing the $\leq$- and $\geq$-concepts, contradicting our assumption.

4 The 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 [10] 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 2 are added to the 6 inference rules

$$\mathbf{7}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L) \qquad \neg L(a)}{C(a)} \qquad \mathbf{8}\ \frac{C \sqcup (L_1 \sqcap L_2 \cdots \sqcap L(a)) \qquad \neg L(a)}{C}$$

$$\mathbf{9}\ \frac{W \sqcup (\leq nR.C) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W(a) \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)} \qquad \mathbf{10}\ \frac{W \sqcup (\leq nR.C)(a) \qquad \{R(a,b_i)\}_{i=1}^{n+1}}{W \sqcup \bigsqcup_{i=1}^{n+1} \neg C(b_i)}$$

Fig. 2. ABox inference rules

presented in Figure 1. As before, the resolved literals have to be maximal in their respective concepts.

An important property of the ABox DL calculus is that the rules for data axioms do not involve $\geq$-concepts. This suggests that all inference steps involving $\geq$-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 $\geq$-concepts are inferred; (2) next we perform the actual data reasoning using a much simpler TBox (as all $\geq$-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 $\geq$-concepts, we can work with function-free clauses, which makes the reasoning task much easier [9].

5   Conclusion

We have presented the DL calculus, a resolution based algorithm for deciding the consistency of a $\mathcal{SHIQ}$ TBox. The novelty of this calculus is that it is defined directly on DL axioms. We hope that further research will reveal that the DL calculus provides a reasonable alternative to the Tableau Method for TBox reasoning.

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 permanent 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., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2004)

2. Haarslev, V., 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. (2004) 163–173

3. Haarslev, V., Möller, R., van der Straeten, R., 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. (2004) 148–157

4. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. **5**(2) (2007) 51–53

5. Hustadt, U., Motik, B., Sattler, U.: Reasoning for Description Logics around SHIQ in a resolution framework. Technical report, FZI, Karlsruhe (2004)

6. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany (January 2006)

7. Lukácsy, G., Szeredi, P., Kádár, B.: Prolog based description logic reasoning. In de la Banda, M.G., Pontelli, E., eds.: Proceedings of 24th International Conference on Logic Programming (ICLP'08),Udine, Italy. (December 2008) 455–469

8. Colmerauer, A., Roussel, P.: The birth of Prolog. ACM, New York, NY, USA (1996)

9. Lukácsy, G., Szeredi, P.: Efficient description logic reasoning in Prolog: the DLog system. Theory and Practice of Logic Programming (in press) (April 2009) http://arxiv.org/abs/0904.0578.

10. Zombori, Z.: Efficient two-phase data reasoning for description logics. In Bramer, M., ed.: IFIP AI. Volume 276 of IFIP., Springer (2008) 393–402

11. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In Robinson, A., Voronkov, A., eds.: Handbook of Automated Reasoning. Volume 1. North Holland (2001) 19–100