# Expressive Description Logic Reasoning Using First-Order Resolution

Zsolt Zombori

Budapest University of Technology and Economics,
Department of Computer Science and Information Theory, Hungary
Tel: 36-70-2651891
zombori@cs.bme.hu

**Abstract.** Description Logic languages are being used more and more frequently for knowledge representation. This creates an increasing demand for efficient automated DL reasoning. Although several DL reasoners are available, they typically do not scale well with the increase of data. In this paper we present a resolution based approach: we show how to transform description logic axioms to a set of function-free clauses of first-order logic. These axioms can afterwards be used for a well scaling, query oriented data reasoning. The transformation is given for the $\mathcal{SHIQ}$ DL language, but we show how to extend the result to a $\mathcal{RIQ}$ DL knowledge base. The method described has been implemented in a module of the DLog reasoner openly available on SourceForge to download.

**Keywords:** Description Logic, ABox reasoning, resolution, SHIQ, RIQ, DLog

## Introduction

Description Logics (DL) is a family of languages designed for conveniently describing domain specific knowledge of various applications. Today it is commonly used hence there is an increasing demand for efficient reasoning in DL. Existing implementations are mostly based on the so called tableau method ([1], [2]) which works just fine for deducing new rules from existing ones, but it can be rather slow when it comes to dealing with large amount of data. In practice, however, the latter situation is becoming more and more typical.
We present a resolution based reasoning algorithm for the $\mathcal{SHIQ}$ DL language. Afterwards we show how to exponentially encode a $\mathcal{RIQ}$ knowledge base into a $\mathcal{ALCHIQ}$ knowledge base, which is $\mathcal{SHIQ}$ without transitivity axioms.
We have developed the DLog system, a DL reasoner which answers large data queries efficiently. This program is prepared for situations in which the relevant data is too much to be loaded into main memory and hence can only be accessed through direct database queries.
The DLog approach breaks the reasoning task into two parts: the first phase works only with the terminology part of the knowledge base and the second phase constitutes the data reasoning. The present paper deals

with the first phase. Although our algorithm is tailored to fit in the DLog reasoner, we argue that the results presented here can be used indepdentently by any other DL data reasoner.
Section 1 gives a brief summary of description logics and first-order resolution as well as a resolution based approach from [3] for theorem proving in $\mathcal{SHIQ}$ DL. Section 2 constitutes our first result: it presents the first phase of DLog, i.e., how to transform the rules of a $\mathcal{SHIQ}$ DL knowledge base into a function-free set of clauses which can be used subsequently for efficient query driven data reasoning. This is achieved by a modification to the algorithm in [3]. Our modified calculus has already been presented at WCC2008 ([4]). Section 3 contains our second result: we show how to transform a $\mathcal{RIQ}$ knowledge base into an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base. Section 4 gives a brief overview of the DLog system which is described in detail in [5]. Finally, in Section 5 we mention some of the work that is still ahead of us.

# 1 Background

This section gives a recollection of some notions necessary to understand the paper and gives references to relevant sources.

## 1.1 Description Logics

Description Logics (DLs) [6] is family of logic languages designed to be a convenient means of knowledge representation. They can be embedded into first-order logic, but – contrary to the latter – they are mostly decidable which gives them a great practical applicability. A DL knowledge base consists of two parts: the TBox (terminology box) and the ABox (assertion box). The TBox contains rules that hold in a specific domain. The ABox stores knowledge about individuals.
The main building blocks of a DL knowledge base are *concepts*, that represent sets of individials and *roles* that represent binary relations, i.e., sets of pairs of individuals. Complex concepts and roles can be built from simpler ones using concept and role constructors: the set of available constructors determines the expressitity of the language and naturally defines a language family. We introduce three languages: $\mathcal{SHIQ}$, $\mathcal{ALCHIQ}$, $\mathcal{RIQ}$.

**$\mathcal{SHIQ}$** The DL language that is probably the best known is $\mathcal{SHIQ}$. It contains expressive language elements, while preserving satisfiability, hence it is a good compromise between expressivity and complexity.

**Definition 1.** *Let $\mathcal{C}$ be a set of concept names and $\mathcal{R}$ a set of role names. The set of roles is $\mathcal{R} \sqcup \{R^- \mid R \in \mathcal{R}\}$. We define the function $Inv$ on roles such that $Inv(R) = R^-$ if $R$ is a role name and $Inv(R^-) = R$.*
*A* role inclusion axiom *(RIA) is an expression of the form $R \sqsubseteq S$, where $R, S$ are roles. A* transitivity axiom *is of the form $Trans(R)$ where $R$ is a role. A* $\mathcal{SHIQ}$-role hierarchy *is a set of role inclusion*

*axioms together with a set of transitivity axioms. For a role hierarchy $\mathcal{R}$ we define $\sqsubseteq^*$ to be a transitive-reflexive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{Inv(R) \sqsubseteq Inv(S) | R \sqsubseteq S \in \mathcal{R}\}$. Role $S$ is called a* sub-role *of $R$ if $S \sqsubseteq^* R$. A role is* simple *if it has no sub-role $S$ such that $Trans(S) \in \mathcal{R}$.*
*The set of $\mathcal{SHIQ}$-concepts is the smallest set such that 1) every concept name is a concept, 2) $\top$ and $\bot$ are concepts and 3) if $C, D$ are concepts, $R$ is a role, $S$ is a simple role and $n$ is a nonnegative integer, then $C \sqcup D$, $C \sqcap D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\leq nS.C$ and $\geq nS.C$ are also concepts.*
*A general concept inclusion axiom (GCI) is an expression of the form $C \sqsubseteq D$ for two $\mathcal{SHIQ}$-concepts $C, D$. A $\mathcal{SHIQ}$-*terminology *is a set of GCIs.*
*Let $\mathcal{I} = \{a, b, c, \dots\}$ be a set of individual names. An* assertion *is of the form $C(a)$, $R(a, b)$, $a = b$ or $a \neq b$ for $a, b \in \mathcal{I}$, a role $R$ and $\mathcal{SHIQ}$-concept $C$. An* ABox *is a set of assertions.*
*A $\mathcal{SHIQ}$ knowledge base $KB$ can be broken into three parts: a terminology ($KB_\mathcal{T}$), a role hierarchy ($KB_\mathcal{R}$) and an ABox ($KB_\mathcal{A}$).*

The semantics of $\mathcal{SHIQ}$ is defined as follows:

**Definition 2.** *An interpretation $I = (\Delta^I, f^I)$ consists of a set $\Delta^I$ called the* domain *of $I$ and a valuation $f^I$ which maps every concept to a subset of $\Delta^I$, every role to a subset of $\Delta^I \times \Delta^I$ and every individual name to a member of $\Delta^I$ such that, for all concepts $C, D$ roles $R, S$ and nonnegative integers $n$, the following equations hold, where $\sharp S$ denotes the cardinality of a set $S$:*

$$
\begin{aligned}
(R^-)^I &= \left\{(x, y) \mid (y, x) \in R^I\right\} \\
\top^I &= \Delta^I \\
\bot^I &= \emptyset \\
(\neg C)^I &= \Delta^I \setminus C^I \\
(C \sqcap D)^I &= C^I \cap D^I \\
(C \sqcup D)^I &= C^I \cup D^I \\
(\forall R.C)^I &= \left\{x \mid \forall y : \langle x, y\rangle \in R^I \rightarrow y \in C^I\right\} \\
(\exists R.C)^I &= \left\{x \mid \exists y : \langle x, y\rangle \in R^I \wedge y \in C^I\right\} \\
(\leq nR.C)^I &= \left\{x \mid \sharp\left\{y \mid \langle x, y\rangle \in R^I \wedge y \in C^I\right\} \leq n\right\} \\
(\geq nR.C)^I &= \left\{x \mid \sharp\left\{y \mid \langle x, y\rangle \in R^I \wedge y \in C^I\right\} \geq n\right\}
\end{aligned}
$$

*An interpretation $I$* satisfies
- *terminology $\mathcal{T}$ if and only if $C^I \subseteq D^I$ for each $C \sqsubseteq D \in \mathcal{T}$. In this case we say that $I$ is a model of $\mathcal{T}$.*
- *role hierarhcy $\mathcal{R}$ if and only if $S^I \subseteq R^I$ for each $S \sqsubseteq R \in \mathcal{R}$ and $R^I$ is transitive for each $Trans(R) \in \mathcal{R}$. In this case we say that $I$ is a model of $\mathcal{R}$.*
- *ABox $\mathcal{A}$ if and only if $a^I \in C^I$ for each $C(a) \in \mathcal{A}$ and $\langle a^I, b^I\rangle \in R^I$ for each $R(a, b) \in \mathcal{A}$, $a^I = b^I$ for each $a = b \in \mathcal{A}$ and $a^I \neq b^I$ for each $a \neq b \in \mathcal{A}$. In this case we say that $I$ is a model of $\mathcal{A}$.*

A knowledge base $KB$ is said to be *satisfiable* in case there exists an interpretation $I$ which is a model of $KB_{\mathcal{T}}$, $KB_{\mathcal{R}}$ and $KB_{\mathcal{A}}$. In this paper, we are interested in the following reasoning task: given a DL knowledge base $KB$ and a query expression Q possibly containing variables. We would like to decide whether Q is a logical consequence of $KB$, i.e., every model $I$ of $KB$ is also a model of Q. If Q contains no variables we expect a yes/no answer. If variables appear in the query, we would like to obtain the complete list of individual names that, when substituted for the variables, result in assertions that follow from $KB$. This task is equivalent to checking if $KB \cup \{\neg Q\}$ is satisfiable. Hence, in the following, we will only address satisfiability checking.

***ALCHIQ*** The following sublanguage of $\mathcal{SHIQ}$ will be important in the rest of the paper:

**Definition 3.** *An $\mathcal{ALCHIQ}$ knowledge base is a $\mathcal{SHIQ}$ knowledge base that contains no transitivity axioms.*

***RIQ*** In $\mathcal{SHIQ}$ we only allowed RIAs of the form $S \sqsubseteq R$ for (possibly inverse) roles $R$ and $S$. Besides, we could declare that some roles are transitive. With transitivity, we can make statements like: *The firend of a friend is a friend* or that *If A is located in B and B is located in C then A is located in C.* However, often it would also be convenient to be able to say things like *The wife of a friend is a friend as well* or that *If A is located in B and B is a subdivision of C, then A is located in C.* It might also be useful in an ontology of family relations to say that *The mother of a spouse is a mother-in-low.* Motivated by these examples, we introduce *generalised RIA*s:

**Definition 4.** *A generalised role inclusion axiom (RIA) is of the form $w \sqsubseteq R$ where $R$ is an atomic role name and $w = S_1 \circ S_2 \circ \ldots S_n$, i.e., $w$ is obtained by composing $n$ roles. A generalised role hierarchy is a set of generalised RIAs.*

In the following, when it leads to no ambiguity, we will indicate composition of roles by simply writing them after each other, i.e., instead of $S_1 \circ S_2$ we will write $S_1 S_2$. Restricting $R$ to atomic roles, is no real restriction and is only meant to make the syntax simpler. Note that the axiom $w \sqsubseteq R$ is equivalent to $Inv(w) \sqsubseteq Inv(R)$, hence we can always choose the one in which the right hand side is an atomic role name. In the presence of generalised RIAs, there is no need for transitivity axioms, since the RIA $RR \sqsubseteq R$ captures the transitivity of $R$.
Introducing generalised RIAs to $\mathcal{SHIQ}$ leads to undecidability in general ([10]). However, we will focus on an important decidable subcase, when the role hierarchy is *regular*:

**Definition 5.** *Let $\prec$ be a strict partial order on roles. A generalised RIA of the form $w \sqsubseteq R$ is $\prec$-regular if*
- *$w = RR$ or*
- *$w = R^-$ or*

– $w = S_1 S_2 \dots S_n$ *and* $S_i \prec R$ *for* $i \in \{1 \dots n\}$ *or*
– $w = R S_1 S_2 \dots S_n$ *and* $S_i \prec R$ *for* $i \in \{1 \dots n\}$ *or*
– $w = S_1 S_2 \dots S_n R$ *and* $S_i \prec R$ *for* $i \in \{1 \dots n\}$

*A generalised role hierarchy is regular is there exists a strict partial order* $\prec$ *such that each RIA is* $\prec$*-regular. The semantics is defined analogously to* $\mathcal{SHIQ}$*, i.e., a model* $\mathcal{I}$ *satisfies a RIA* $w \sqsubseteq R$ *if* $w^{\mathcal{I}} \subseteq R^{\mathcal{I}}$*. The Description Logic* $\mathcal{RIQ}$ *is obtained from* $\mathcal{SHIQ}$ *by replacing role hierarchies and transitivity axioms with regular role hierarchies.*

*With the change of the role hierarchy, the definition of simple roles changes as well:*

– *Every role name that does not occur on the right hand side of a RIA is simple.*
– *A role name* $R$ *is simple if, for each RIA* $w \sqsubseteq R, w = S$ *for some simple role* $S$*.*
– *An inverse role* $S^-$ *is simple if* $S$ *is simple.*

### 1.2 Resolution

Resolution [7] is a complete method for proving satisfiability of first-order clauses. Its two inference rules are summarised in Figure 1 where $\sigma$ is the most general unifier of $B$ and $C$ ($\sigma = \mathrm{MGU}(B, C)$).

$$\frac{A \vee B \qquad \neg C \vee D}{A\sigma \vee D\sigma} \qquad\qquad \frac{A \vee B \vee C}{A\sigma \vee C\sigma}$$

**Fig. 1.** Binary Resolution and Positive Factoring

Ordered resolution [8] refines this technique by imposing an order in which the literals of a clause have to be resolved. This reduces the search space while preserving completeness. It is parametrised with an *admissible ordering* ($\succ$) on literals and a *selection function*.

Basic superposition [9] is an extension of ordered resolution which has explicit inference rules for handling equality. The rules are summarised in Figure 2, where $E|_p$ is a subexpression of $E$ in position $p$, $E[t]_p$ is the expression obtained by replacing $E|_p$ in $E$ with $t$, $C$ and $D$ denote clauses, $A$ and $B$ denote literals without equality and $E$ is an arbitrary literal. The necessary conditions for the applicability of each rule are given in the following list:

**Hyperresolution**: *(i)* $\sigma$ is the most general unifier such that $A_i\sigma = B_i\sigma$, *(ii)* each $A_i\sigma$ is maximal in $C_i\sigma$, and there is no selected literal in $(C_i \vee A_i)\sigma$, *(iii)* either every $\neg B_i$ is selected, or $n = 1$ and nothing is selected and $\neg B_1\sigma$ is maximal in $D\sigma$.

**Positive factoring**: *(i)* $\sigma = \mathrm{MGU}(A, B)$, *(ii)* $A\sigma$ is maximal in $C\sigma$ and nothing is selected in $A\sigma \vee B\sigma \vee C\sigma$.

**Equality factoring**: *(i)* $\sigma = \mathrm{MGU}(s, s')$, *(ii)* $t\sigma \not\succ s\sigma$, *(iii)* $t'\sigma \not\succ s'\sigma$, *(iv)* $(s = t)\sigma$ is maximal in $(C \vee s' = t')\sigma$ and nothing is selected in $(C \vee s = t \vee s' = t')\sigma$.

| | |
|---|---|
| Hyperresolution | $\frac{(C_1 \vee A_1) \ldots (C_n \vee A_n) \quad (D \vee \neg B_1 \vee \cdots \vee \neg B_n)}{(C_1 \vee \cdots \vee C_n \vee D)\sigma}$ |
| Positive factoring | $\frac{A \vee B \vee C}{A\sigma \vee C\sigma}$ |
| Equality factoring | $\frac{C \vee s = t \vee s' = t'}{(C \vee t \neq t' \vee s' = t')\sigma}$ |
| Reflexivity resolution | $\frac{C \vee s \neq t}{C\sigma}$ |
| Superposition | $\frac{(C \vee s = t) \quad (D \vee E)}{(C \vee D \vee E[t]_p)\sigma}$ |

**Fig. 2.** Inference rules of Basic Superposition

**Reflexivity resolution**: *(i)* $\sigma = \mathrm{MGU}(s, t)$, *(ii)* in $(C \vee s \neq t)\sigma$ either $(s \neq t)\sigma$ is selected or nothing is selected and $(s \neq t)\sigma$ is maximal in $C\sigma$.
**Superposition**: *(i)* $\sigma = \mathrm{MGU}(s, E|_p)$, *(ii)* $t\sigma \not\succ s\sigma$, *(iii)* if $E = {}'w = v'$ and $E|_p$ is in $w$ then $v\sigma \not\succ w\sigma$ and $(s\sigma = t\sigma) \not\succ (w\sigma = v\sigma)$, *(iv)* $(s = t)\sigma$ is maximal in $C\sigma$ and nothing is selected in $(C \vee s = t)\sigma$, *(v)* in $(D \vee E)\sigma$ either $E\sigma$ is selected or nothing is selected and $E\sigma$ is maximal, *(vi)* $E|_p$ is not a variable position.

An important feature of basic superposition is that it remains complete even if we do not allow superposition into variables or terms substituted for variables. For this reason we keep track of such positions, by surrounding them with '[ ]' and refer to them as *variable positions* or *marked positions*. So, for instance, applying substitution $\sigma = \{x/g(y)\}$ to clause $C = R(x, y) \vee P(x)$ results in $C\sigma = R([g(y)], y), P([g(y)])$.

### 1.3 Resolution Based Reasoning for DL

In [3] a resolution based theorem proving algorithm for the $\mathcal{SHIQ}$ DL language is presented. In the first step, transitivity axioms are eliminated, at the expense of adding some new GCIs. The obtained $\mathcal{ALCHIQ}$ knowledge base is not logically equivalent to the original one, however [3] proves that the two knowledge bases are equisatisfiable.
In the following definition, *NNF(C)* denotes the negation normal form of $C$, i.e., negation is pushed inwards to atomic concepts.

**Definition 6.** *For a $\mathcal{SHIQ}$ knowledge base KB,* clos(KB) *denotes the smallest set of concepts that satisfies the following conditions:*

- *if $C \sqsubseteq D \in KB$, then NNF($\neg C \sqcup D$) $\in clos(KB)$;*
- *if $C \equiv D \in KB$, then NNF($\neg C \sqcup D$) $\in clos(KB)$ and NNF($\neg D \sqcup C$) $\in clos(KB)$;*
- *if $C(a) \in KB$ then NNF($C$) $\in clos(KB)$;*
- *if $C \in clos(KB)$ and $D$ is a subconcept of $C$ then $D \in clos(KB)$;*
- *if $\leq nR.C \in clos(KB)$ then NNF($\neg C$) $\in clos(KB)$;*
- *if $\forall R.C \in clos(KB)$, $S \sqsubseteq^* R$, and $Trans(S) \in KB_{\mathcal{R}}$, then $\forall S.C \in clos(KB)$.*

*We call $clos(KB)$ the* concept closure of KB.

**Definition 7.** *For any $\mathcal{SHIQ}$ DL knowledge base $KB$, $\Omega(KB)$ is an $\mathcal{ALCHIQ}$ knowledge base constructed as follows:*
- *$\Omega(KB)_{\mathcal{R}}$ is obtained from $KB_{\mathcal{R}}$ by removing all axioms $Trans(R)$;*
- *$\Omega(KB)_{\mathcal{T}}$ is obtained by adding to $KB_{\mathcal{T}}$ for each concept $\forall R.C \in clos(KB)$ and role $S$ such that $S \sqsubseteq^* R$ and $Trans(S) \in KB_{\mathcal{R}}$ the axiom $\forall R.C \sqsubseteq \forall S.(\forall S.C)$;*
- *$\Omega(KB)_{\mathcal{A}} = KB_{\mathcal{A}}$*

**Proposition 1.** *$KB$ is satisfiable if and only if $\Omega(KB)$ is satisfiable.*

*Proof.* See [3]. □

After eliminating transitivity axioms, the knowledge base, together with the negation of the query is transformed into a set of first-order clauses with a characteristic structure. These are referred to as $\mathcal{ALCHIQ}$ clauses and are summarised in Figure 3, where:
- $\mathbf{P}$(t) is a possibly empty disjunction $(\neg)P_1(t) \vee \cdots \vee (\neg)P_n(t)$ of unary literals;
- $\mathbf{P(f}$(x)): is a possibly empty disjunction $\mathbf{P_1}(f_1(x)) \vee \cdots \vee \mathbf{P_n}(f_n(x))$;
- t is a term that is surely not marked;
- [t] is a term that is surely marked;
- <t> is a term that may or may not be marked;
- $\# \in \{=, \neq\}$;

**Fig. 3.** $\mathcal{ALCHIQ}$ clauses

$$\neg R(x,y) \vee S(y,x) \tag{1}$$

$$\neg R(x,y) \vee S(x,y) \tag{2}$$

$$\mathbf{P}(x) \vee R(x, < f(x) >) \tag{3}$$

$$\mathbf{P}(x) \vee R([f(x)], x) \tag{4}$$

$$\mathbf{P_1}(x) \vee \mathbf{P_2}(< \mathbf{f}(x) >) \vee \bigvee (< f_i(x) > \# < f_j(x) >) \tag{5}$$

$$\mathbf{P_1}(x) \vee \mathbf{P_2}([g(x)]) \vee \mathbf{P_3}(< \mathbf{f}([g(x)]) >) \bigvee (< t_i > \# < t_j >) \tag{6}$$

where $t_i$ and $t_j$ are of the form $f([g(x)])$ or of the form $x$

$$\mathbf{P_1}(x) \vee \bigvee_{i=1}^{n} (\neg R(x, y_i) \vee \bigvee_{i=1}^{n} \mathbf{P_2}(y_i) \vee \bigvee_{i,j=1}^{n \times n} (y_i = y_j) \tag{7}$$

$$\mathbf{R}(< a >, < b >) \vee \mathbf{P}(< t >) \vee \bigvee (< t_i > \# < t_j >) \tag{8}$$

where $t, t_i$ and $t_j$ are either a constant or a term $f_i([a])$

The reasoning task is reduced to deciding whether the obtained first-order clauses are satisfiable. This is answered using basic superposition

extended with a method called *decomposition*. [3] shows that the set of $\mathcal{ALCHIQ}$ clauses is bounded and that any inference with premises taken from a subset N of $\mathcal{ALCHIQ}$ results in either *(i)* an $\mathcal{ALCHIQ}$ clause or *(ii)* a clause redundant in N[1] or *(iii)* a clause that can be decomposed to, i.e., substituted with two $\mathcal{ALCHIQ}$ clauses without affecting satisfiability. These results guarantee that the saturation of an $\mathcal{ALCHIQ}$ set terminates.

### 1.4 Separating TBox and ABox Reasoning

The drawback of the resolution algorithm outlined above is that it can be painfully slow. In general, resolution with saturation is a bottom-up strategy and computes all logical consequences of the clause set, many of which are irrelevant to deciding our question. It would be nice to be able to use some more efficient, query oriented, top-down mechanism. Unfortunately, such mechanisms are available only for more restrictive languages, such as Horn Clauses. One can get around this problem by breaking the reasoning into two tasks: first perform a saturation based preprocessing to deduce whatever could not be deduced otherwise and then use a fast top-down reasoner.
Note that complex reasoning is required because of the rules (TBox) of the knowledge base and that in a typical real life situation there is a relatively small TBox and a large ABox. Furthermore, the rules in the TBox are likely to remain the same over time while the ABox data can change continuously. Hence we would like to move forward all inferences involving the TBox only, perform them separately and then let the fast reasoner (whatever that will be) do the data related steps when a query arrives.
In the framework of basic superposition, when more than one inference steps are applicable, we are free to choose an order of execution, providing a means to achieve the desired separation. Elements from the ABox appear only in clauses of type (8). [3] gives two important results about the role of ABox axioms in the saturation process:

**Proposition 2.** *An inference from $\mathcal{ALCHIQ}$ clauses results in a conclusion of type* (8) *if and only if there is a premise of type* (8)*.*

**Proposition 3.** *A clause of type* (8) *cannot participate in an inference with a clause of type* (4) *or* (6)*.*

In light of Proposition 2, we can move forward ABox independent reasoning by first performing all inference steps involving only clauses of type (1) – (7). [3] calls this phase the saturation of the TBox. Afterwards, Proposition 3 allows us to eliminate clauses of type (4) and (6). Besides making the clause set smaller, this elimination is crucial because in the remaining clauses there can be no function symbol embedded into another (this only occurred in clauses of type (6)). The importance of this result comes out in the second phase of the reasoning, because the

[1] A redundant clause is a special case of other clauses in N and can be removed.

available top down mechanisms are rather sensitive to the presence of function symbols.
By the end of the first phase DL reasoning has been reduced to deciding the satisfiability of first-order clauses of type (1) - (3), (5), (7) and (8), where every further inference involves at least one premise of type (8). For the second phase, i.e., data reasoning, [3] uses a datalog engine which requires function-free clauses. Therefore (unary) functional relations are transformed to new binary predicates and new constant names are added: for each constant $a$ and each function $f$ the new constant $a_f$ is introduced to represent $f(a)$. Note that this transformation requires processing the whole ABox.

## 2 Towards Pure Two-Phase Reasoning

In this section we introduce modifications to the course of basic superposition saturation of $\mathcal{ALCHIQ}$ clauses. We do this to be able to perform more inferences before accessing the ABox. This is not just a mere regrouping of tasks, we will see that the algorithm produces a crucially simpler input for the second phase with a huge impact on its performance efficiency and on the available data reasoning algorithms. The improvement is achieved by eliminating function symbols from the clauses derived from the TBox.

### 2.1 Where Do Functions Come From?

The $\mathcal{SHIQ}$ DL knowledge base that we started from contained no functions. Then, after translating TBox axioms to first-order logic we had to eliminate existential quantifiers using skolemisation which introduced new function symbols. The ABox remained function-free, hence everything that is to know about the functions is contained in the TBox. This means we should be able to perform all function-related reasoning before accessing the ABox.

### 2.2 The Modified Calculus

We modify basic superposition presented in Subsection 1.2 by altering the necessary conditions to apply each rule. The new conditions are given below, with the newly added conditions underlined:

**HyperresolutionTBox**: *(i)* $\sigma$ is the most general unifier such that $A_i\sigma = B_i\sigma$, *(ii)* each $A_i\sigma$ is maximal in $C_i\sigma$, and either there is no selected literal in $(C_i \vee A_i)\sigma$ or $A_i$ contains a function symbol, *(iii)* either every $\neg B_i$ is selected, or $n = 1$ and $\neg B_1\sigma$ is maximal in $D\sigma$ *(iv)* none of the premises contain constants.

**HyperresolutionABox**: *(i)* $\sigma$ is the most general unifier such that $A_i\sigma = B_i\sigma$, *(ii)* each $A_i\sigma$ is maximal in $C_i\sigma$, and there is no selected literal in $(C_i \vee A_i)\sigma$, *(iii)* either every $\neg B_i$ is selected, or $n = 1$ and nothing is selected and $\neg B_1\sigma$ is maximal in $D\sigma$, *(iv)* each $A_i$ is ground, *(v)* $D\sigma$ is function-free.

**Positive factoring**: *(i)* $\sigma = \text{MGU}(A, B)$, *(ii)* $A\sigma$ is maximal in $C\sigma$ and either nothing is selected in $A\sigma \vee B\sigma \vee C\sigma$ or $A$ contains a function symbol.
**Equality factoring**: *(i)* $\sigma = \text{MGU}(s, s')$, *(ii)* $t\sigma \not\succ s\sigma$, *(iii)* $t'\sigma \not\succ s'\sigma$, *(iv)* $(s = t)\sigma$ is maximal in $(C \vee s' = t')\sigma$ and either nothing is selected in $C\sigma$ or $s = t \vee s' = t'$ contains a function symbol.
**Reflexivity resolution**: *(i)* $\sigma = \text{MGU}(s, t)$, *(ii)* in $(C \vee s \neq t)\sigma$ either $(s \neq t)\sigma$ is selected or $s \neq t$ contains a function symbol or nothing is selected and $(s \neq t)\sigma$ is maximal in $C\sigma$.
**Superposition**: *(i)* $\sigma = \text{MGU}(s, E|_p)$, *(ii)* $t\sigma \not\succ s\sigma$, *(iii)* if $E = {}'w = v'$ and $E|_p = w|_{p'}$ then $v\sigma \not\succ w\sigma$ and $(s\sigma = t\sigma) \not\succ (w\sigma = v\sigma)$, *(iv)* $(s = t)\sigma$ is maximal in $C\sigma$ and either nothing is selected in $(C \vee s = t)\sigma$ or $s = t$ contains a function symbol, *(v)* in $(D \vee E)\sigma$ either $E\sigma$ is selected or nothing is selected and $E\sigma$ is maximal, *(vi)* $E|_p$ is not a variable position.

Note that hyperresolution is broken into two rules (HyperresolutionTBox and HyperresolutionABox) which differ only in the necessary conditions. In the following by *original calculus* we refer to the basic superposition presented in Subsection 1.2 and by *modified calculus* we mean the rules of basic superposition with the restrictions listed above. We will prove that the new calculus can be used to solve the reasoning task.

**Proposition 4.** *The modified calculus remains correct and complete.*

*Proof.* The inference rules of basic superposition are all valid even if we do not impose any restrictions on their applicability. Since in the new calculus only the conditions are altered, it remains correct.
The modifications that weaken the requirements to apply a rule only extend the deducible set of clauses, so they do not affect completeness.
In case of hyperresolution, let us first consider only the new condition *(iv)* and disregard condition *(v)* on HyperresolutionABox. A hyperresolution step in its original form has a main premise of type (7), some (possibly zero) side premises of type (3) – (4) and some (possibly zero) side premises of type (8). This one step can be broken into two by first resolving the main premise with all side premises of type (3) and (4) (by one HyperresolutionTBox inference step) and then resolving the rest of selected literals with side premises of type (8) (applying a HyperresolutionABox step). A hyperresolution step in the original calculus can be replaced by two steps in the modified one, so completeness is preserved.
All that remains to be proved is that condition *(v)* on HyperresolutionABox does not prevent completeness. For this, let us consider a refutation in the original calculus that uses a hyperresolution step. If all side premises are of type (3) and (4) then it can be substituted with a HyperresolutionTBox step. Similarly, if all side premises are of type (8), then we can change it to HyperresolutionABox, as clauses of type (7) are function-free, satisfying condition *(v)*. The only other option is that there are both some premises of type (3) and of type (8)[2]. The result of

[2] It is shown in [3] that clauses of type (8) and (4) participating in an inference result in a redundant clause so we need not consider this case.

such step is a clause of the following type:

$$\mathbf{P_1}(x) \vee \bigvee \mathbf{P_2}(a_i) \vee \bigvee \mathbf{P_2}([f_i(x)]) \vee$$
$$\vee \bigvee (a_i = a_j) \vee \bigvee ([f_i(x)] = [f_j(x)]) \vee \bigvee ([f_i(x)] = a_j)$$

At some point each function symbol is eliminated from the clause (by the time we reach the empty clause everything gets eliminated). In the modified calculus we will be able to build an equivalent refutation by altering the order of the inference steps: we first apply HyperresolutionTBox which introduces all the function symbols, but none of the constants, then we bring forward the inference steps that eliminate function symbols and finally we apply HyperresolutionABox. The intermediary steps between HyperresolutionTBox and HyperresolutionABox are made possible by the weakening of the corresponding necessary conditions. Notice, that by the time HyperresolutionABox is applied, functions are eliminated so condition *(v)* is satisfied.
We conclude that for any proof tree in the original calculus we can construct a proof tree in the modified calculus, so the latter is complete. □

**Proposition 5.** *Saturation of a set of $\mathcal{ALCHIQ}$ clauses using the modified calculus terminates.*

*Proof.* (sketch) We build on the results in [3], that a $\mathcal{SHIQ}$ knowledge base can be transformed into first-order clauses of type (1) – (8) and that clauses of type (8) are of the form $C(a), R(a,b), \neg S(a,b), a = b$ or $a \neq b$, i.e., initially they do not contain any function symbols. We will also use the fact that in the original calculus any inference with premises taken from a subset N of $\mathcal{ALCHIQ}$ results in either *(i)* an $\mathcal{ALCHIQ}$ clause or *(ii)* a clause redundant in N or *(iii)* a clause that can be substituted with two $\mathcal{ALCHIQ}$ clauses via decomposition.
All modifications (apart from breaking hyperresolution into two) affect clauses having both function symbols and selected literals, in that we can resolve with the literal containing the function symbol before eliminating all selected literals. Such a clause can only arise as a descendant of a HyperresolutionTBox step. After applying HyperresolutionTBox, we can obtain the following clauses:

$$\mathbf{P_1}(x) \vee \bigvee (\neg R(x, y_i)) \vee \bigvee \mathbf{P_2}(y_i) \vee \bigvee \mathbf{P_2}([f_i(x)]) \vee \tag{9}$$
$$\vee \bigvee (y_i = y_j) \vee \bigvee ([f_i(x)] = [f_j(x)]) \vee \bigvee ([f_i(x)] = y_j)$$

In the following, it will be comfortable for us to consider a clause set that is somewhat broader than (9), in which function symbols can appear in inequalities as well. This set is:

$$\mathbf{P_1}(x) \vee \bigvee (\neg R(x, y_i)) \vee \bigvee \mathbf{P_2}(y_i) \vee \bigvee \mathbf{P_2}([f_i(x)]) \vee \tag{10}$$
$$\vee \bigvee (y_i = y_j) \vee \bigvee (< f_i(x) > \# < f_j(x) >) \vee \bigvee (< f_i(x) > \# \; y_j)$$

where $\# \in \{=, \neq\}$. Of course, every clause of type (9) is of type (10) as well.

Let us see what kind of inferences can involve clauses of type (10). First, it can be a superposition with a clause of type (3) or (5). In the case of (3) the conclusion is decomposed (in terms of [3]) into clauses of type (3) and (10), while in the case of (5) we obtain a clause of type (10). Second, we can resolve clauses of type (10) with clauses of type (10) or (5). The conclusion is of type (10). Finally, we can apply HyperresolutionABox with some side premises of the form $R(a, b_i)$, but notice that only if the literals with function symbols are missing. The result is of type (8). This means that during saturation, we will only produce clauses of type (1) – (8) and (10).
It is easy to see that there can only be a limited number of clauses of type (10) over a finite signature. Hence the modified calculus will only generate clauses from a finite set, so the saturation will terminate. □

### 2.3 Implementing Two-Phase Reasoning

We will use the modified calculus to solve the reasoning task in two phases. Our separation differs from that of [3] in that function symbols are eliminated during the first phase, without any recourse to the ABox. Our method is summarised in Algorithm 1, where steps (1) - (3) constitute the first phase of the reasoning and step (4) is the second phase, i.e., the data reasoning.

**Algorithm 1** $\mathcal{SHIQ}$ reasoning

1. We transform the $\mathcal{SHIQ}$ knowledge base to a set of clauses of types (1) - (8), where clauses of type (8) are function-free.
2. We saturate the TBox clauses (types (1) - (7)) with the modified calculus. The obtained clauses are of type (1) - (7) and (10).
3. We eliminate all clauses containing function symbols.
4. We add the ABox clauses (type (8)) and saturate the set.

To show that our method is adequate, we first formulate the following proposition:

**Proposition 6.** *A function-free ground clause can only be resolved with function-free clauses. Furthermore, the resolvent is ground and function-free.*

*Proof.* It follows simply from the fact that a constant $a$ cannot be unified with a term $f(x)$ and from condition *(v)* on HyperresolutionABox. □

We are now ready to state our main claim:

**Theorem 1.** *Algorithm 1 is a correct, complete and finite $\mathcal{SHIQ}$ DL theorem prover.*

*Proof.* We know from Proposition 5 that saturation with the modified calculus terminates. After saturating the TBox, every further inference

will have at least one premise of type (8), because the conclusions inferred after this point are of type (8) (Proposition 6). From this follows, (using Proposition 6) that clauses with function symbols will not participate in any further steps, hence they can be removed. In light of this and taking into account that the modified calculus is correct and complete (Proposition 4), so is Algorithm 1. □

By the end of the first phase of reasoning, we obtain clauses of the following types:

$$\neg R(x, y) \vee S(y, x) \tag{11}$$

$$\neg R(x, y) \vee S(x, y) \tag{12}$$

$$\mathbf{P}(x) \tag{13}$$

$$\mathbf{P_1}(x) \vee \bigvee_i (\neg R(x, y_i)) \vee \bigvee_i \mathbf{P_2}(y_i) \vee \bigvee_{i,j} (y_i = y_j) \tag{14}$$

$$(\neg) R(a, b) \tag{15}$$

$$C(a) \tag{16}$$

$$a = b \tag{17}$$

$$a \neq b \tag{18}$$

We have completely eliminated function symbols and are now ready to start the data reasoning.

### 2.4 Benefits of Eliminating Functions

The following list gives some advantages of eliminating function symbols before accessing the ABox.

1. It is more **efficient**. Whatever ABox independent reasoning we perform after having accessed the data will have to be repeated for every possible substitution of variables.
2. It is **safer**. A top-down reasoner that has to be prepared for arguments containing function symbols is very prone to fall into infinite loops. Special attention needs to be paid to ensure the reasoner does not generate goals with ever increasing number of function symbols.
3. We get **equality handling** for free. In the resulting TBox only clauses of type (14) contain equality that can be eliminated by a mere check whether two constants from the ABox refer to the same object which is usually well known by the creators of the database. Note that equality treatment in general makes the reasoning task much more complex. This is why we had to use basic superposition.
4. ABox reasoning without functions is **qualitatively easier**. Some algorithms, such as those for datalog reasoning, are not available in the presence of function symbols. We have seen in Section 1.4 that [3] solves this problem by syntactically eliminating functions, but this has two drawbacks: first, equality reasoning is required (an introduced constant might be equal to an ABox constant) and second, this transformation requires scanning through the whole ABox, which might not be feasible when we have a lot of data.

### 2.5 Summary of the modified calculus

In this section we have presented a saturation algorithm that can be used to transform a $\mathcal{SHIQ}$ TBox to a set of function-free clauses. The transformation is independent of the ABox, and hence of the size of the ABox. It can be seen as a preprocessing for ABox reasoning and hence any resolution based ABox reasoning algorithm can make use of it. The main benefit is that without functions the ABox reasoning can be more focused, i.e., less sensitive to the size of the ABox.

## 3 Transforming $\mathcal{RIQ}$ to $\mathcal{ALCHIQ}$

In this section we examine $\mathcal{RIQ}$, i.e., the extension of $\mathcal{SHIQ}$ with regular role inclusion axioms (RIAs) and show that such RIAs can be transformed into equisatisfiable $\mathcal{ALCHIQ}$ concepts. Most of the definitions that will be introduced are based on ([10]), which gives a tableau procedure for deciding $\mathcal{RIQ}$.
For each role $R$, the authors define a non-deterministic finite automaton (NFA) that captures the role paths that are subsumed by $R$. These automata are used during the construction of a tableau, to "keep track" of role paths. In the remaining of this section we show that the automata can be used to transform the initial $\mathcal{RIQ}$ knowledge base to an equisatisfiable $\mathcal{ALCHIQ}$ knowledge base. The main benefit is that the treatment of the role hierarchy becomes independent of the tableau algorithm. Hence, any algorithm that decides satisfiability for an $\mathcal{ALCHIQ}$ knowledge base can be used for satisfiability checking of a $\mathcal{RIQ}$ knowledge base. In particular, the resolution calculus presented in Section 2.2 is applicable. This result extends the input language of the DLog reasoner from $\mathcal{SHIQ}$ to $\mathcal{RIQ}$.

### 3.1 Building automata to represent RIAs

In this subsection we define a scheme for constructing finite automata to represent regular role hierarchies. We use the same construction as presented in [10].

**Definition 8.** *Let $\mathcal{R}$ be a regular role hierarchy. For each role name $R$ occurring in $\mathcal{R}$, the NFA $A_R$ is defined as follows: $A_R$ contains a single initial state $i_R$ and a single final state $f_R$ with the transition $i_R \xrightarrow{R} f_R$. Moreover, for each $w \sqsubseteq R \in \mathcal{R}$, $A_R$ contains the following transitions:*

1. *if $w = RR$, then $A_R$ contains $f_R \xrightarrow{\epsilon} i_R$,*
2. *if $w = S_1 \dots S_n$ and $S_1 \neq R \neq S_n$, then $A_R$ contains $i_R \xrightarrow{\epsilon} i_w \xrightarrow{S_1} f_w^1 \xrightarrow{S_2} f_w^2 \dots \xrightarrow{S_n} f_w^n \xrightarrow{\epsilon} f_R$*
3. *if $w = RS_1 \dots S_n$ then $A_R$ contains $f_R \xrightarrow{\epsilon} i_w \xrightarrow{S_1} f_w^1 \xrightarrow{S_2} f_w^2 \dots \xrightarrow{S_n} f_w^n \xrightarrow{\epsilon} f_R$*
4. *if $w = S_1 \dots S_n R$ then $A_R$ contains $i_R \xrightarrow{\epsilon} i_w \xrightarrow{S_1} f_w^1 \xrightarrow{S_2} f_w^2 \dots \xrightarrow{S_n} f_w^n \xrightarrow{\epsilon} i_R$*

*where all $f^i_w, i_w$ are assumed to be distinct.*
*Next, we introduce mirrored copies of automata, where all transitions go backwards and the initial and final states are switched. Formally, in the mirrored copy of an NFA we carry out the following modifications:*

- *final states are made non-final but intial*
- *initial states are made non-initial but final*
- *each transition $p \xrightarrow{S} q$ is replaced with transition $q \xrightarrow{Inv(S)} p$*
- *each transition $p \xrightarrow{\epsilon} q$ is replaced with transition $q \xrightarrow{\epsilon} p$.*

*We define NFAs $\hat{A_R}$ as follows:*

- *if $R^- \sqsubseteq R \notin \mathcal{R}$ then $\hat{A_R} := A_R$.*
- *if $R^- \sqsubseteq R \in \mathcal{R}$ then $\hat{A_R}$ is obtained as follows: first, take the disjoint union of $A_R$ with a mirrored copy of $A_R$. Second, make $i_R$ the only initial state, $f_R$ the only final state. Finally, for $f'_R$ the copy of $f_R$ and $i'_R$ the copy of $i_R$, add transitions $i_R \xrightarrow{\epsilon} f'_R, f'_R \xrightarrow{\epsilon} i_R, i'_R \xrightarrow{\epsilon} f_R$ and $f_R \xrightarrow{\epsilon} i'_R$.*

*Afterwards, the NFAs $B_R$ are defined inductively over $\prec$:*

- *if $R$ is minimal w.r.t. $\prec$, then we set $B_R := \hat{A_R}$.*
- *otherwise, $B_R$ is the disjoint union of $\hat{A_R}$ with a copy $B'_S$ of $B_S$ for each transition $p \xrightarrow{S} q$ in $\hat{A_R}$ with $S \neq R$. Moreover, for each such transition, we add $\epsilon$-transitions from $p$ to the intial state in $B'_S$ and from the final state of $B'_S$ to $q$, and we make $i_R$ the only initial state and $f_R$ the only final state in $B_R$.*

*Finally, the automaton $B_{R^-}$ is a mirrored copy of $B_R$.*

**Proposition 7.** *For each role $R \in \mathcal{R}$ the size of $B_R$ is bounded exponentially in the size of $\mathcal{R}$.*

*Proof.* See [10]. □

**Definition 9.** *We denote by $B_R(q, *)$ the automaton that differs from $B_R$ only in its initial state, which is $q$. Analogously, $B_R(*, q)$ differs from $B_R$ only in its final state, which is $q$.*

**Proposition 8.** *For a regular role hierarchy $\mathcal{R}$ and interpretation $I$, $I$ is a model of $\mathcal{R}$ if and only if, for each (possibly inverse) role $S$ occurring in $\mathcal{R}$, each word $w \in L(B_S)$ and each $\langle x, y\rangle \in w^{\mathcal{I}}$, we have $\langle x, y\rangle \in S^{\mathcal{I}}$.*

*Proof.* See [10]. □

Proposition 8 states that two individuals are $S$-connected exactly when there is a role path $w$ between them accepted by $B_S$. This result gives us a key to handle value restrictions. Suppose individual $x$ satisfies some $S$-restriction. If this is a maximum restriction ($\leq kS.C$), then $S$ must be a simple role and the restriction effects only the immediate neighbours of $x$. This case is already treated in $\mathcal{SHIQ}$. If it is a minimum restriction ($\geq kS.C$), the restriction can be made true by adding some $S$-successors to $x$. The only problematic case is universal restriction ($\forall S.C$), because finding all $S$-successors might be rather difficult. However, Proposition 8 tells us that it is the role paths described by $B_S$ that we need to check to look for $S$-successors.

### 3.2 A Motivating Example

Before formally defining the transformation of automata generated from the role hierarchy into axioms, we try to give an intuition through a small example. Suppose the role hierarchy of a knowledge base consists of the single axiom

$$PQ \sqsubseteq R$$

where $R, P, Q$ are role names. One of the things that this axiom tells us is that in case an individual $x$ satisfies $\forall R.C$ for some concept $C$, then the individuals connected to $x$ through a $P \circ Q$ chain have to be in $C$. This consequence can be described easily by the following GCI:

$$\forall R.C \sqsubseteq \forall P.\forall Q.C$$

or equivalently, we can introduce new concept names to avoid too much nesting of complex concepts:

$$\begin{aligned} \forall R.C &\sqsubseteq X_1 \\ X_1 &\sqsubseteq \forall P.X_2 \\ X_2 &\sqsubseteq \forall Q.C \end{aligned}$$

Of course, these axioms only provide for the correct propagation of concept $C$ and a new set of similar axioms is required for all other concepts. However, we only need to consider the universal restrictions that appear as subconcepts of some axiom in the knowledge base. These concepts can be determined by a quick scan of the initial knowledge base. For example, if the TBox contains the following GCIs:

$$\begin{aligned} D &\sqsubseteq \forall R.C \\ \top &\sqsubseteq \forall R.D \end{aligned}$$

then, only concepts $C$ and $D$ appear in the scope of a universal $R$-restriction. Let us add a copy of the above CGIs for both $C$ and $D$ and eliminate the role hierarchy. We obtain the following TBox:

$$\begin{aligned} D &\sqsubseteq \forall R.C & \top &\sqsubseteq \forall R.D \\ \forall R.C &\sqsubseteq X_1 & \forall R.D &\sqsubseteq Y_1 \\ X_1 &\sqsubseteq \forall P.X_2 & Y_1 &\sqsubseteq \forall P.Y_2 \\ X_2 &\sqsubseteq \forall Q.C & Y_2 &\sqsubseteq \forall Q.D \end{aligned}$$

The two knowledge bases have different signatures and hence have different models, however they are equisatisfiable. We will prove this by showing that one can construct from the model of one knowledge base a model of the other.

### 3.3 Translating automata to concept inclusion axioms

In this subsection we formally define the transformation of a regular role hierarchy into GCIs. In the end we obtain from a $\mathcal{RIQ}$ knowledge base an $\mathcal{ALCHIQ}$ knowledge base. We make use of the notion of concept closure ($clos(KB)$) provided in Definition 6. The transformation itself is analogous to how transitivity axioms were eliminated from $\mathcal{SHIQ}$ (Definition 7). Here, the situation is more complex as we have to take into consideration more sophisticated role paths.
For each concept $\forall R.C \in clos(KB)$ and each automaton state $s$ of $B_R$, we intoduce a new concept name $X_{(s,R,C)}$. The concepts associated with the initial and final states of $B_R$ are denoted with $X_{(start,R,C)}$ and $X_{(stop,R,C)}$, respectively.

**Definition 10.** *For any $\mathcal{RIQ}$ DL knowledge base $KB$, $\Omega(KB)$ is an $\mathcal{ALCHIQ}$ knowledge base constructed as follows:*

- *$\Omega(KB)_\mathcal{R}$ is obtained from $KB_\mathcal{R}$ by removing all RIAs $w \sqsubseteq R$ such that $R$ is not simple;*
- *$\Omega(KB)_\mathcal{T}$ is obtained by adding to $KB_\mathcal{T}$ for each concept $\forall R.C \in clos(KB)$ the following axioms:*
  1. *$\forall R.C \sqsubseteq X_{(start,R,C)}$*
  2. *$X_{(p,R,C)} \sqsubseteq X_{(q,R,C)}$ for each $p \xrightarrow{\epsilon} q \in B_R$*
  3. *$X_{(p,R,C)} \sqsubseteq \forall S.X_{(q,R,C)}$ for each $p \xrightarrow{S} q \in B_R$*
  4. *$X_{(stop,R,C)} \sqsubseteq C$*
- *$\Omega(KB)_\mathcal{A} = KB_\mathcal{A}$*

**Proposition 9.** *The size of $\Omega(KB)$ is bounded exponentially in the size of $KB$.*

*Proof.* We know from Proposition 7 that the size of each $B_R$ is bounded exponentially in the size of $KB_\mathcal{R}$ and consequently in the size of $KB$. So for each concept $\forall R.C \in clos(KB)$ we intoduce at most exponentially many new GCIs of type 1-4. The size of $clos(KB)$ is linear in $KB$, so the total number of GCIs introduced is at most exponential in the size of $KB$. □

The following proposition will be useful for proving that $KB$ and $\Omega(KB)$ are equisatisfiable.

**Proposition 10.** *Let $KB$ be some $\mathcal{RIQ}$ knowledge base and $I$ be a model of $\Omega(KB)$. Assume that $\alpha \in (\forall R.C)^I$ and there is some $\beta$ and role path $w \in L(B_R)$ such that $\langle \alpha, \beta \rangle \in w^I$. Then $\beta \in C^I$.*

*Proof.* Let $w = S_1 S_2 \dots S_n$, where $S_i$ is possibly an $\epsilon$ transition. Let $start = b_0, b_1, \dots b_n = stop$ be states of $B_R$ along the $w$ path. Since $\langle \alpha, \beta \rangle \in w^I$, there are individuals $\alpha = a_0, a_1, \dots a_n = \beta$ such that $\langle a_{i-1}, a_i \rangle \in S^I$. Note that in case $S_i = \epsilon$ then $a_{i-1} = a_i$.
We show inductively that $a_i \in X^I_{(b_i,R,C)}$ for all $0 \leq i \leq n$. For this we use the axioms added in the construction of $\Omega(KB)$. The axiom of type 1 ensures that the base case holds: $\alpha \in X^I_{(start,R,C)}$, i.e., $a_0 \in X^I_{(b_0,R,C)}$. For the inductive step, suppose first that $S_i$ is an $\epsilon$ transition. Then

$a_i = a_{i-1}$. By the inductive hypothesis $a_{i-1} \in X^I_{(b_{i-1},R,C)}$, and the corresponding axiom of type 2 ensures that $a_i \in X^I_{(b_i,R,C)}$. In the other case, when $S_i$ is not an $\epsilon$ transition, the same argument referring to a corresponding axiom of type 3 ensures that $a_i \in X^I_{(b_i,R,C)}$.
Hence we know that $a_n \in X^I_{(b_n,R,C)}$, i.e., $\beta \in X^I_{(stop,R,C)}$. This, together with the axiom of type 4 ensures that $\beta \in C^I$. □

We are ready to formulate the main claim of this section:

**Theorem 2.** *$KB$ is satisfiable if and only if $\Omega(KB)$ is satisfiable.*

*Proof.* ($\Rightarrow$) Let $I$ be a model of $KB$. We extend this model to an interpretation $I'$ of $\Omega(KB)$. $I'$ differs from $I$ only in the interpretation of the new concepts $X_{(s,R,C)}$:

$$X^{I'}_{(s,R,C)} = \left\{ y \mid \exists x (x \in (\forall R.C)^I \wedge (\exists w \in L(B_R(*,s))(\langle x,y\rangle \in w^{\mathcal{I}}))) \right\}$$

We prove that $I'$ is a model of $\Omega(KB)$, by showing that the axioms added in the definition of $\Omega(KB)$ are true. We consider the four cases separately:

1. $\forall R.C \sqsubseteq X_{(start,R,C)}$
   Suppose $y \in (\forall R.C)^{I'}$. Then, by choosing $x = y$ and $w = \epsilon$, we can apply the above definition to show that $y \in X^{I'}_{(start,R,C)}$.
2. $X_{(p,R,C)} \sqsubseteq X_{(q,R,C)}$
   Suppose $y \in X^{I'}_{(p,R,C)}$. Then, there is some $x \in (\forall R.C)^{I'}$ and some $w \in L(B_R(*,p))$ such that $\langle x,y\rangle \in w^{\mathcal{I}'}$. Since $p \xrightarrow{\epsilon} q \in B_R$, it also holds that $w \in L(B_R(*,q))$. Hence, the same $x$ and $w$ testify that $y \in X^{I'}_{(q,R,C)}$.
3. $X_{(p,R,C)} \sqsubseteq \forall S.X_{(q,R,C)}$
   Suppose $y \in X^{I'}_{(p,R,C)}$. Then, there is some $x \in (\forall R.C)^{I'}$ and some $w \in L(B_R(*,p))$ such that $\langle x,y\rangle \in w^{\mathcal{I}'}$. Let $z$ be some $S^{I'}$-successor of $y$, i.e., $\langle y,z\rangle \in S^{\mathcal{I}'}$. Since $p \xrightarrow{S} q \in B_R$, it also holds that $wS \in L(B_R(*,q))$. Hence, $x$ and $wS$ testify that $z \in X^{I'}_{(q,R,C)}$. This holds for all $S^{I'}$-successors of $y$, hence $y \in \forall S.X^{I'}_{(q,R,C)}$.
4. $X_{(stop,R,C)} \sqsubseteq C$
   Suppose $y \in X^{I'}_{(stop,R,C)}$. Then, there is some $x \in (\forall R.C)^{I'}$ and some $w \in L(B_R)$ such that $\langle x,y\rangle \in w^{\mathcal{I}'}$. Since $I$ and $I'$ only differ in the extension of new concepts, we also have $x \in (\forall R.C)^I$ and $\langle x,y\rangle \in w^{\mathcal{I}}$. From the latter, we infer using Proposition 8 that $\langle x,y\rangle \in R^{\mathcal{I}}$. Since $x \in (\forall R.C)^I$, it follows that $y \in C^I$ and from that we conclude that $y \in C^{I'}$.

($\Leftarrow$) Let $I$ be a model of $\Omega(KB)$ and $I'$ an interpretation constructed from $I$ as follows:

- $\Delta^{I'} = \Delta^I$;
- For each individual $a$, $a^{I'} = a^I$;
- For each atomic concept $A \in clos(KB)$, $A^{I'} = A^I$;
- For each role $R$, $R^{I'} = \left\{ \langle x,y\rangle \mid \exists w \in L(B_R)(\langle x,y\rangle \in w^I) \right\}$

By construction and referring to Proposition 8, $I'$ satisfies the role hierarchy $KB_{\mathcal{R}}$. Since $R \in L(B_R)$, we have $R^I \subseteq R^{I'}$. Furthermore, if $R$ is simple then $R^I = R^{I'}$.

For concepts in $clos(KB)$, we define the strict partial order $\lhd$: $C \lhd D$ if and only if $C$ or NNF($C$) occur in $D$. We will use induction on $\lhd$ to show that for each $D \in clos(KB)$, $D^I \subseteq D^{I'}$. For the base case, i.e., when $D$ is an atomic concept or a negated atomic concept, this follows immediately from the definition of $I'$. We now turn to the inductive step:

- For $D = C_1 \sqcap C_2$, assume that $\alpha \in (C_1 \sqcap C_2)^I$ for some $\alpha$. Then, $\alpha \in C_1^I$ and $\alpha \in C_2^I$. By the inductive hypothesis, $\alpha \in C_1^{I'}$ and $\alpha \in C_2^{I'}$, so $\alpha \in (C_1 \sqcap C_2)^{I'}$.
- For $D = C_1 \sqcup C_2$, assume that $\alpha \in (C_1 \sqcup C_2)^I$ for some $\alpha$. If $\alpha \in C_1^I$, then by induction we also have $\alpha \in C_1^{I'}$; if $\alpha \in C_2^I$, then by induction we also have $\alpha \in C_2^{I'}$. Either way, $\alpha \in (C_1 \sqcup C_2)^{I'}$.
- For $D = \exists R.C$, assume that $\alpha \in (\exists R.C)^I$. Then, $\beta$ exists such that $\langle \alpha, \beta \rangle \in R^I$ and $\beta \in C^I$. By induction, $\beta \in C^{I'}$. Since $R^I \subseteq R^{I'}$, we have $\langle \alpha, \beta \rangle \in R^{I'}$, so $\alpha \in (\exists R.C)^{I'}$.
- For $D = (\geq nR.C)$, assume that $\alpha \in (\geq nR.C)^I$. Then, there are at least $n$ distinct domain elements $\beta_i$ such that $\langle \alpha, \beta_i \rangle \in R^I$ and $\beta_i \in C^I$. By induction, $\beta_i \in C^{I'}$. Since $R^I \subseteq R^{I'}$, we have $\langle \alpha, \beta_i \rangle \in R^{I'}$, so $\alpha \in (\geq nR.C)^{I'}$.
- For $D = (\leq nR.C)$, we have $R^I = R^{I'}$ since $R$ is simple. Let $E =$ NNF($\neg C$). Assume that $\alpha \in (\geq nR.C)^I$, but $\alpha \notin (\geq nR.C)^{I'}$. Then, there exists $\beta$ such that $\langle \alpha, \beta \rangle \in R^I, \beta \notin C^I, \beta \in C^{I'}$, i.e., $\beta \in E^I$ and $\beta \notin E^{I'}$. However, since $E \in clos(KB)$, by induction we have $\beta \in E^{I'}$, which is a contradiction. Hence, $\alpha \in (\leq nR.C)^{I'}$.
- For $D = \forall R.C$, assume that $\alpha \in (\forall R.C)^I$, but $\alpha \notin (\forall R.C)^{I'}$. Then some $\beta$ exists such that $\langle \alpha, \beta \rangle \in R^{I'}$ and $\beta \notin C^{I'}$. By the definition of $R^{I'}$ there is some $w \in L(B_R)$ such that $\langle \alpha, \beta \rangle \in w^I$. Using Proposition 10, it follows that $\beta \in C^I$. By induction, $C^I \subseteq C^{I'}$, so $\beta \in C^{I'}$, which is a contradiction. Hence $\alpha \in (\forall R.C)^{I'}$.

□

### 3.4 Summary of the $\mathcal{RIQ}$ to $\mathcal{ALCHIQ}$ transformation

In this section we defined a transformation $\Omega$ that maps an arbitrary $\mathcal{RIQ}$ knowledge base to an $\mathcal{ALCHIQ}$ knowledge base. Theorem 2 states that the transformation preserves satisfiability. We also showed that the transformation increases the size of the TBox with at most an exponential factor (Proposition 9). This is assymptotically optimal: $\mathcal{ALCHIQ}$ is known to be ExpTime-hard while $\mathcal{RIQ}$ is 2ExpTime-hard ([11]), so $\mathcal{RIQ}$ is indeed exponentially harder than $\mathcal{ALCHIQ}$.

Using this result, any algorithm that decides satisfiability for $\mathcal{ALCHIQ}$ can decide satisfiability for $\mathcal{RIQ}$. In particular, the modified calculus persented in Subsection 2.2 is applicable.

## 4 The DLog System

The DLog system is a DL data reasoner, written in the Prolog language, which implements a two-phase reasoning algorithm, based on first-order resolution.
In the first phase the TBox of the input knowledge base is transformed into a set of clauses of type (11) – (14). The transformation uses the saturation algorithm described in [3] and Section 2.2, hence it fully supports the $\mathcal{SHIQ}$ language. We are in the process of extending DLog from $\mathcal{SHIQ}$ to $\mathcal{RIQ}$ by implementing the transformation $\Omega$ descibed in Section 3.
The clauses obtained from the TBox after the first phase are used to build a Prolog program. It is the execution of this program – run with an adequate query – that performs the second phase, i.e., the data reasoning. The second phase is focused in that it starts out from the query and only accesses parts of the ABox that are relevant to answering the query. The relevant part is determined by the clauses derived from the TBox. Hence, the performance of DLog is not affected by the presence of irrelevant data. Futhermore, the ABox can be accessed through direct database queries and needs not be stored in memory. To our best knowledge, DLog is the only DL reasoner which does not need to scan through the whole ABox. Thanks to this, DLog can be used to reason over really large amounts of data stored in external databases.
The transformation to Prolog uses the PTTP approach ([12]), a complete theorem prover technology for first-order logic. Details about the obtained Prolog program and the architecture of the DLog system can be found in [5].
The program is in exerimental stage. We implemented all the reasoning algorithms and we have prototype implementations for various futher features, such as support for ABoxes stored in database. In the near future we plan to incorporate all our results in a reasoner that proves useful for the DL community. An earlier, stable version of DLog that supports $\mathcal{SHIQ}$ is available at `http://dlog-reasoner.sourceforge.net`.

## 5 Future Work

As noted in the previous section, while all important reasoning algorithms are implemented, the DLog is not yet ready to use. One of the most urgent tasks is providing the system with an adequate interface. Currently, we only support the DIG ([13]) format for the input knowledge base and query. We would like to provide the system with an OWL interface (see [14] and [15]). Moreover, we would like to incorporate database support into the reasoner. Once these tasks are done, we need to do thorough testing to evaluate the DLog with respect to other DL resoners such as RacerPro, Pellet, Hermit, KAON2.
On the theoretical side, we are curious to see how far we can extend the expressivity of DLog beyond $\mathcal{RIQ}$, approximating, as much as possible $\mathcal{SROIQ}(\mathcal{D})$, the language behind OWL2 ([15]). Our efforts to include

nominals, i.e., concepts that hold a single individual, have not been successful. Nominals blur the distinction between TBox and ABox (in the extreme, the whole ABox can be represented as TBox axioms when nominals are allowed), so it is unlikely that we could break the reasoning into two phases such that the first phase is independent of the ABox. Other extensions, such as concrete domains, are still mostly unexplored.

## Summary

This paper has two contributions. First, we showed how to extend the results in [3] to transform a $\mathcal{SHIQ}$ TBox into a set of first-order clauses. The obtained clauses are of a rather simple structure, namely they are function-free. This paves the way for fast, query oriented inference algorithms to perform data reasoning tasks originally formulated in DL. Second, we described a transformation that maps $\mathcal{RIQ}$ knowledge bases into equisatifiable $\mathcal{ALCHIQ}$ knowledge bases. The two results put together provide a preprocessing phase for efficient $\mathcal{RIQ}$ data reasoning. The DLog system implements the algorithms described in the paper. However, the transformations are not "DLog specific" and we believe that our results can be useful for other DL reasoners as well.

## Acknowledgements

The work reported in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME" project. This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009

## References

1. Hladik, J., Model, J., Theory, C.F.A., Dresden, T.: Tableau systems for SHIO and SHIQ
2. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic SHIQ (2000)
3. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
4. Zombori, Z.: Efficient two-phase data reasoning for description logics. In Bramer, M., ed.: IFIP AI. Volume 276 of IFIP., Springer (2008) 393–402
5. Lukácsy, G., Szeredi, P.: Efficient description logic reasoning in Prolog: the DLog system. Theory and Practice of Logic Programming **09**(03) (May 2009) 343–414
6. Horrocks, I.: Reasoning with expressive description logics : Theory and practice. Language (2002) 1–15
7. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1) (1965) 23–41

8. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In Robinson, J.A., Voronkov, A., eds.: Handbook of Automated Reasoning. Elsevier and MIT Press (2001) 19–99
9. Bachmair, L., Ganzinger, H.: Strict basic superposition. Lecture Notes in Computer Science **1421** (1998) 160–174
10. Horrocks, I., Sattler, U.: Decidability of SHIQ with complex role inclusion axioms. In Gottlob, G., Walsh, T., eds.: IJCAI, Morgan Kaufmann (2003) 343–348
11. Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: In Proc. KR08. (2008)
12. Stickel, M.E.: A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog. Theor. Comput. Sci. **104**(1) (1992) 109–128
13. Bechhofer, S.: The DIG Description Logic Interface: DIG/1.1. In: Proceedings of the 2003 Description Logic Workshop. (2003) `http://dl-web.man.uk/dig/2003/02/interface.pdf`.
14. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: the making of a web ontology language. Web Semantics: Science, Services and Agents on the World Wide Web **1**(1) (2003) 7 – 26
15. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Web Semant. **6** (November 2008) 309–322