Querying expressive DLs

Sergio Tessaris
The university of Manchester, Manchester UK
Email: tessaris@cs.man.ac.uk

1 Introduction

A serious shortcoming of many Description Logic based knowledge representation systems is the inadequacy of their query languages. In this paper we show how the query answering technique presented in [7] for $\mathcal{ALC}$ can be extended for a DL featuring role restrictions like transitivity and role hierarchy.

Recent years have witnessed the transfer of algorithmic techniques used for terminological reasoning to the development of both algorithms and (optimised) implementations that also support Abox reasoning (see [4, 10]). Although these systems provide sound and complete Abox reasoning for very expressive logics, their utility is limited by their very weak Abox query languages. Typically, these only support instantiation (is an individual i an instance of a concept C), realisation (what are the most specific concepts i is an instance of) and retrieval (which individuals are instances of C). The reason for this weakness is that, in these expressive logics, all reasoning tasks are reduced to that of determining KB satisfiability (consistency). In particular, instantiation is reduced to KB (un)satisfiability by transforming the query into a negated assertion; however, this technique cannot be used (directly) for queries involving roles because these logics do not support role negation.

However, in [2] and [8] is shown that a more sophisticate reduction to KB (un)satisfiability can be used for answering conjunctive queries similar to those supported by relational databases.[1] This query language allows complex Abox structures (e.g., cyclical structures) to be retrieved by using variables to enforce co-reference.

In this paper we focus on the DL $\mathcal{SH}f$, showing that the technique used in [2] and [8] can be adapted to this logic as well. Lack of space forces us to omit most of the details, which can be found in [9]. Our goal is providing the reader the intuition of the problems that a DL like $\mathcal{SH}f$ poses to the already presented technique, and how the algorithm can be modified to provide correct and complete answers.

We focus on the problem of answering boolean queries, i.e., determining if a query is true with respect to a KB. Retrieval can be turned into a set of boolean queries for all candidate tuples.

[1] It is inspired by the use of Abox reasoning to decide conjunctive query containment (see [6, 1]).

2 Preliminaries

The description logic $\mathcal{SHf}$ is an extension of the logic $\mathcal{ALC}$ to include transitive roles, role hierarchy and functional restriction.

The DL $\mathcal{SHf}$ is built over a signature of distinct concept ($\mathcal{CN}$), role ($\mathcal{RN}$) and individual ($\mathcal{O}$) sets of names. In addition, we distinguish two non–overlapping subsets of $\mathcal{RN}$ ($\mathcal{TRN}$ and $\mathcal{FRN}$) which denote the transitive and the functional roles. The set of all $\mathcal{SHf}$ concepts is the smallest set such that every concept name in $\mathcal{CN}$ and the symbols $\top$, $\bot$ are concepts, and if C and D are concepts and R a role name in $\mathcal{RN}$, then $\neg C$, $(C \sqcap D)$, $(C \sqcup D)$, $(\forall R.C)$, and $(\exists R.C)$ are concepts.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty domain $\Delta^{\mathcal{I}}$ and a interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps concepts into subsets of $\Delta^{\mathcal{I}}$, individual names into elements of $\Delta^{\mathcal{I}}$, and role names into subsets of the cartesian product of $\Delta^{\mathcal{I}}$ ($\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). Concept names are interpreted as subsets of $\Delta^{\mathcal{I}}$, while complex expressions are interpreted according to the equations

$$
\begin{array}{ll}
\top^{\mathcal{I}} = \Delta^{\mathcal{I}} & (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
\bot^{\mathcal{I}} = \emptyset & (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
 & (\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}
\end{array}
$$

In addition, the interpretation function must satisfy the transitive and functional restrictions on role names; i.e. for any $R \in \mathcal{TRN}$ if $(x, y) \in R^{\mathcal{I}}$ and $(y, z) \in R^{\mathcal{I}}$, then $(x, z) \in R^{\mathcal{I}}$, and for any $F \in \mathcal{FRN}$ if $(x, y) \in F^{\mathcal{I}}$ and $(x, z) \in F^{\mathcal{I}}$, then $y = z$.

DL knowledge bases A $\mathcal{SHf}$ knowledge base $\mathbf{K}$ is a finite set of statements of the forms:

$$C \sqsubseteq D, R \sqsubseteq S, a{:}C, \langle a, b\rangle{:}R$$

where C, D are $\mathcal{SHf}$ concepts, R, S role names, and a, b individual names. The first two kinds of statement are called *terminological*, while the two latter ones are called *assertional*. Intuitively, terminological statements describe intensional properties of all the elements of the domain, while assertional statements assign properties of some named elements.

We say that an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies the KB statement $C \sqsubseteq D$ ($R \sqsubseteq S$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$), and the statement $a{:}C$ ($\langle a, b\rangle{:}R$) iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$). When an interpretation $\mathcal{I}$ satisfies a statement α, we use the notation $\mathcal{I} \models \alpha$. An interpretation $\mathcal{I}$ satisfies (or is a model for) a KB $\mathbf{K}$ iff it satisfies all the statements in $\mathbf{K}$ (written as $\mathcal{I} \models \mathbf{K}$).

Query language Typical DL systems provide a very limited language for querying knowledge bases, which is often restricted to verifying the satisfiability of a given concept, and to checking whether an individual name is a member of a concept (or instantiation). These reasoning services can be seen as the process of verifying whether a given statement is a logical consequence of the knowledge base (written as $\mathbf{K} \models \alpha$). The meaning of logical consequence is given in terms of interpretations, and $\mathbf{K} \models \alpha$ iff for any interpretation $\mathcal{I}$, $\mathcal{I} \models \mathbf{K}$ implies $\mathcal{I} \models \alpha$. For example, instantiation can be expressed by $\mathbf{K} \models a{:}C$.

Using the same mechanism we extend the kind of queries we can ask by introducing a *conjunctive query language* whose terms are assertional statements presented in the previous section (see [7]). For this purpose we consider a set of variable names $\mathcal{V}$ distinct from the names previously introduced. Analogously to conjunctive queries in the database setting, variables can be used in place of individuals and are considered as existentially quantified.

A DL conjunctive query is defined as a set of terms of the form $x{:}C$ or $\langle x, y\rangle{:}R$, where C is a concept, R is a role name, and x, y are variable or individual names taken from $\mathcal{V} \cup \mathcal{O}$. We call the terms of the first kind *concept terms* and the ones of the latter one *role terms*.

The semantics of a conjunctive query follows the very same schema shown for the knowledge bases in the last paragraph, with the difference that we need to consider the variable names; because the satisfiability of a term may be affected by the assignment of the variables. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we consider *evaluations* defined as a mappings of elements in $\mathcal{V}$ to the interpretation domain $\Delta^{\mathcal{I}}$.[2] We say that the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies the term $x{:}C$ w.r.t. an evaluation ν (written as $\mathcal{I} \models_\nu x{:}C$) iff $\nu(x) \in C^{\mathcal{I}}$, and analogously for a term like $\langle x, y\rangle{:}R$. This is extended to arbitrary conjunctive queries: an interpretation $\mathcal{I}$ satisfies the conjunctive query $q = \{t_1, \ldots, t_n\}$ w.r.t. an evaluation ν iff $\mathcal{I} \models_\nu t$ for any $t \in q$.

We push the query language one step further, by introducing a weak sort of disjunction into the picture. A DL *disjunctive query* is a set of conjunctive queries $Q = \{q_1, \ldots, q_n\}$; with the restriction that variables are not shared among different disjuncts.[3] An interpretation $\mathcal{I}$ satisfies the query Q w.r.t. an evaluation ν iff there is at least a q in Q such that $\mathcal{I} \models_\nu q$.

We are not really interested in the evaluation itself but only on the satisfiability of the given query. Then, we say that $\mathcal{I}$ satisfies the query Q (written $\mathcal{I} \models Q$) iff there is an evaluation ν such that $\mathcal{I} \models_\nu Q$.

[2] In fact, we consider individuals as well as variables in the evaluations (i.e. $(\mathcal{V} \cup \mathcal{O}) \rightarrow \Delta^{\mathcal{I}}$) Imposing that assignment and the interpretation function agree on individual names.

[3] In other words, variables are existentially quantified in front of each disjunct; i.e. $(\exists x.q_1) \vee (\exists x.q_2)$ instead of $\exists x.(q_1 \vee q_2)$.

3 Query answering technique

By query answering we intend the automated process of deciding whether a given disjunctive query is logical consequence of the knowledge base. The underlying technique consists in a sophisticated reduction of query answering to the problem of verifying whether a knowledge base is satisfiable (see [7]). This reduction has been first used in [1] for solving the problem of conjunctive query containment for the DL $\mathcal{DLR}$.

The main idea is to consider a conjunctive query as a directed graph, where the nodes are variable and individual names, while the edges are role terms. In addition, concept and role terms provide labels for nodes and edges respectively. A disjunctive query is represented as set of graphs.

For example, the query $\{x{:}\texttt{Start}, \langle x, y\rangle{:}\texttt{Path}, \langle y, z\rangle{:}\texttt{Path}, \langle x, z\rangle{:}\texttt{Path}\}$ corresponds to the graph

$$\begin{tikzcd} & y \arrow[dr, "\texttt{Path}"] & \\ \texttt{Start}\;\; x \arrow[ur, "\texttt{Path}"] \arrow[rr, "\texttt{Path}"'] & & z \end{tikzcd} \tag{1}$$

A set of graph transformation rules is applied to the graphs in the query until all the disjuncts "collapse" into graphs containing a single node. Note that the single node graph corresponds to a conjunctive query containing a single concept term; therefore the collapsed disjunctive query has the form:

$$\mathbf{K} \models \{\{x_1{:}C_1\}, \dots, \{x_n{:}C_n\}\}.$$

When $x_1, \dots, x_n$ are variable names, verifying whether the query is a logical consequence of $\mathbf{K}$ is equivalent to checking the unsatisfiability of the new KB[4]

$$\mathbf{K} \cup \{\top \sqsubseteq (\neg C_1 \sqcap \ldots \sqcap \neg C_n)\}.$$

As shown in [7] the transformation rules may add new statements to the KB and/or add new disjuncts in the query. Since our purpose is to show what happens when the underlying DL language is $\mathcal{SH}f$ instead of $\mathcal{ALC}$, we do not enter into the details here. However, we summarise the key points underlying the transformation rules.

- "Leaf" variables are eliminated by using the existential construct; e.g. $\{\langle x, y\rangle{:}R, y{:}C\}$ becomes $\{x{:}\exists R.C\}$. This procedure is called *rolling up.*

[4]If one or more x_i are individual names the reduction is slightly different (see [7]) but conceptually the same.

- Individual names are eliminated by using the so called *representative* concepts, which are newly introduced concept names associated to individuals; e.g. $\{\langle x, a\rangle{:}R, a{:}C\}$ becomes $\{x{:}\exists R.(P_a \sqcap C)\}$, and at the same time a new statement $a{:}P_a$ is added to the knowledge base in the left hand side.
- Since neither $\mathcal{ALC}$ nor $\mathcal{SH}f$ allow the use of inverse roles, "inverse" rolling up is performed by introducing a new concept name; e.g. $\{\langle x, y\rangle{:}R, x{:}C\}$ becomes $\{y{:}P_{R^-.C}\}$, where $P_{R^-.C}$ is a newly introduced concept name defined by the statement $C \sqsubseteq \forall R.P_{R^-.C}$ in the knowledge base.

Using the ideas explained above it is not difficult to see that tree–shaped queries can be collapsed by getting rid of variable and individual names. The crucial point is how to eliminate variable names appearing in cycles.[5]

This problem can be solved by noticing that $\mathcal{ALC}$ has the tree–model property (see [11]), which ensures that whenever an $\mathcal{ALC}$ formula is satisfiable (w.r.t. a terminology) there is a tree–shaped model for it. This property can be extended to KBs containing assertional statements. In this case the model is a forest where the roots correspond to individual names, and they are arbitrarily interconnected according to assertional statements.

It can be shown that this property is valid not only for KB satisfiability, but it is preserved to the problem of logical implication for disjunctive queries.[6] This enables us to conclude that cycles in queries can be satisfied only if they consist only of individual names.

For example, the knowledge base

$$\{a{:}\texttt{Start}, \langle a, b\rangle{:}\texttt{Path}, \langle b, c\rangle{:}\texttt{Path}, \langle a, c\rangle{:}\texttt{Path}\}$$

satisfies Query (1). The procedure for verifying the satisfiability arbitrarily picks one of the variables in the cycle, and substitute it with any available individual name. If the the chosen variable is z, the result of this step followed by the rolling up using the representative concepts is a disjunctive query with three different disjuncts:

$$\begin{tikzcd} & y\ \exists\texttt{Path}.P_a \\ x\ \texttt{Start} \sqcap \exists\texttt{Path}.P_a \arrow[ur, "\texttt{Path}"] & \end{tikzcd} \quad \begin{tikzcd} & y\ \exists\texttt{Path}.P_b \\ x\ \texttt{Start} \sqcap \exists\texttt{Path}.P_b \arrow[ur, "\texttt{Path}"] & \end{tikzcd} \quad \begin{tikzcd} & y\ \exists\texttt{Path}.P_c \\ x\ \texttt{Start} \sqcap \exists\texttt{Path}.P_c \arrow[ur, "\texttt{Path}"] & \end{tikzcd}$$

Once completed the rolling up, the resulting disjunctive query is

$$\left\{ \begin{array}{l} \{x{:}\texttt{Start} \sqcap (\exists\texttt{Path}.P_a) \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.P_a)\}\,, \\ \{x{:}\texttt{Start} \sqcap (\exists\texttt{Path}.P_b) \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.P_b)\}\,, \\ \{x{:}\texttt{Start} \sqcap (\exists\texttt{Path}.P_c) \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.P_c)\} \end{array} \right\},$$

[5] Individual names are not a problem because the representative concept technique can be used to break the cycle.

[6] Note that this latter point is not trivial, because we cannot assume the reduction of logical implication to satisfiability.

and the KB to be checked for unsatisfiability becomes

$$\left\{\begin{array}{l} a{:}\mathtt{Start}, \langle a,b\rangle{:}\mathtt{Path}, \langle b,c\rangle{:}\mathtt{Path}, \langle a,c\rangle{:}\mathtt{Path}, \\ a{:}P_a, b{:}P_b, c{:}P_c, \\ \top \sqsubseteq \neg(\mathtt{Start} \sqcap (\exists\mathtt{Path}.P_a) \sqcap (\exists\mathtt{Path}.\exists\mathtt{Path}.P_a)), \\ \top \sqsubseteq \neg(\mathtt{Start} \sqcap (\exists\mathtt{Path}.P_b) \sqcap (\exists\mathtt{Path}.\exists\mathtt{Path}.P_b)), \\ \top \sqsubseteq \neg(\mathtt{Start} \sqcap (\exists\mathtt{Path}.P_c) \sqcap (\exists\mathtt{Path}.\exists\mathtt{Path}.P_c)) \end{array}\right\}.$$

In the next section we concentrate on the treatment of cycles in the reduction, showing that the increased expressivity of the DL $\mathcal{SH}f$ requires a more careful treatment of variables in cycles.

4 Cycles: behind the intuition

As mentioned above, the real problem in reducing query answering to KB satisfiability lies on potential cycles in the query. To understand whether and how variables occurring in cycles can be eliminated we consider the "pattern" represented by a query graph. An interpretation satisfies the query if the query graph can be matched against the interpretation itself.

If we can restrict ourselves to interpretations of a well defined class, we can exploit the characteristics of the class to design the transformation rules. What we are after is a completeness result for a class of interpretations $\mathcal{T}$ w.r.t. the problem of logical implication. Given the nature of DLs we should concentrate on tree–like structures, which have the advantage of confining potential cycles to few well defined categories.[7]

Formally, proving correctness and completeness of the class $\mathcal{T}$ correspond to show that for any given knowledge base $\mathbf{K}$, disjunctive query Q, and interpretation $\mathcal{I}$: $\mathcal{I} \models \mathbf{K}$ implies $\mathcal{I} \models Q$ iff for every interpretation $\mathcal{I}' \in \mathcal{T}$, $\mathcal{I}' \models \mathbf{K}$ implies $\mathcal{I}' \models Q$. The "only if" direction is trivial because interpretations in $\mathcal{T}$ are among all the possible interpretations; therefore we concentrate on the "if" direction. In [9] we show the validity of the latter direction by defining a function $\sigma_{\mathbf{K}}$ mapping arbitrary interpretations satisfying a KB $\mathbf{K}$ to interpretations belonging to $\mathcal{T}$. This function satisfies the condition that for any disjunctive query Q and interpretation $\mathcal{I}$ satisfying $\mathbf{K}$, if $\sigma_{\mathbf{K}}(\mathcal{I}) \models Q$ then $\mathcal{I} \models Q$ as well. Once defined this mapping $\sigma_{\mathbf{K}}$, showing the completeness of $\mathcal{T}$ results in a simple chain of implications.

The key of the whole process is choosing the right class $\mathcal{T}$ according to the underlying DL. We decided to define the interpretation domains of elements of $\mathcal{T}$ as sets of nonempty sequences of elements taken from some "alphabet"

[7] We use the adjective "tree–like" because we cannot completely eliminate cycles; e.g. assertional statements can force arbitrary patterns.

domain. Then we used the interpretation of roles for shaping the interpretation as a forest over these sequences.

The idea is that roots of the forest are sequences containing a single element, and individual names are mapped to roots only. Then the tree structures are build by connecting a sequence (by the interpretation of roles) only to sequences equal to the same sequence concatenated to a new element.

For a DL like $\mathcal{ALC}$ we can impose interpretations in $\mathcal{T}$ a strong forest–like structure.

Definition 1 *Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\mathcal{I}$ is a member of $\mathcal{T}$ only if for arbitrary sequences α, β in $\Delta^{\mathcal{I}}$:*

1. *if the pair $(\alpha, \beta) \in P^{\mathcal{I}}$ for some role name P then β is equal to α concatenated to an element of the alphabet domain, or both α and β correspond to individual names;*
2. *if the pair $(\alpha, \beta) \in P^{\mathcal{I}} \cap R^{\mathcal{I}}$ for some different role names P, R then both α and β correspond to individual names.*

We call interpretations satisfying the above restrictions *shrub* interpretations, because they are composed by structures having a core of interconnected elements, to which the trees are hanged.

It is not difficult to show that given the above restrictions, cyclical queries can only be satisfied by identifying variables occurring in cycles with individual names. For convenience, in query graphs we distinguish between *simple* cycles where two nodes in the query are connected by different roles (e.g. like $\langle x, y\rangle{:}P \wedge \langle x, y\rangle{:}R$), and *complex* cycles for all the other kind of cycles. The first restriction of Definition 1 can be used as an argument in the cases of simple cycles, while the second restriction can be used for complex cycles.

The story is more complicate if the underlying DL is enriched with constructs that affect roles; transitivity and role hierarchy in the case of $\mathcal{SH}f$. The two assumptions we made in Definition 1 are no longer valid for $\mathcal{SH}f$. For example, transitivity forces the presence of shortcuts, making the interpretation no longer tree–like and invalidating the first assumption. On the other hand, role hierarchy can be seen as a sort of role conjunction operator, therefore the same pair of elements of the domain can be forced to be connected by more than a role, and this invalidates the second assumption.

In [7] we identified the so called *false cycles* as special cases in which apparently cyclical queries can be satisfied by non–cyclical interpretations. The expressivity of $\mathcal{SH}f$ introduces new troublesome cases in which some kind of cycles may be forced by terminological statements alone; i.e. without the involvement of assertional statements.

For example, if the terminology imposes the transitivity of that the role `Path` then the KB $\{a{:}(\texttt{Start} \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.\top))\}$ satisfies Query (1). However, the described procedure would not detect it. In fact, the KB to be tested would be

$$\left\{ \begin{array}{l} a{:}(\texttt{Start} \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.\top)), a{:}P_a, \\ \top \sqsubseteq \neg(\texttt{Start} \sqcap (\exists\texttt{Path}.P_a) \sqcap (\exists\texttt{Path}.\exists\texttt{Path}.P_a)), \end{array} \right\}$$

and it is satisfiable; as witnessed by the interpretation $\Delta^{\mathcal{I}} = \{1, 2, 3\}$, $a^{\mathcal{I}} = 1$, $\texttt{Start}^{\mathcal{I}} = P_a{}^{\mathcal{I}} = \{1\}$, and $\texttt{Path}^{\mathcal{I}} = \{(1, 2), (2, 3), (1, 3)\}$.

On top of that, the DL $\mathcal{SH}f$ features functional roles which can interact with role hierarchy, making the treatment of simple cycles even more complicate. To illustrate this interaction let us consider F_1, F_2 be two functional roles and R, S, T three different roles s.t. $R \sqsubseteq F_1$, $S \sqsubseteq F_1$, $S \sqsubseteq F_2$, and $T \sqsubseteq F_2$. Given an arbitrary interpretation $\mathcal{I}$, and two pairs (u, v) in $R^{\mathcal{I}}$ and (u, w) in $T^{\mathcal{I}}$, whether or not v and w must be the same actually depends on the existence of an S–successor of u. If there is an element u' s.t. $(u, u') \in S^{\mathcal{I}}$, then both (u, v) and (u, u') are in $(F_1)^{\mathcal{I}}$ and $v = u'$ because of the functionality of $(F_1)^{\mathcal{I}}$; analogously, we can show that $u' = w$, therefore $v = w$. On the other hand, if such an S–successor u' of u does not exist, nothing is forcing v and w to be the same.

In [9] we present a class of interpretations whose members are still forest–like shaped as the ones above mentioned for $\mathcal{ALC}$, but they take into account the peculiarities of $\mathcal{SH}f$. We show that this class is complete w.r.t. the problems of KB satisfiability and disjunctive query answering. Moreover, the described class is used to devise a reduction of disjunctive query answering to KB satisfiability.

We call the interpretations in this class *quasi–transitive shrub* interpretations. The relevant properties of a quasi–transitive shrub $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ can be summarised in the following points (for more details see [9]).

1. Given two arbitrary sequences α, β in $\Delta^{\mathcal{I}}$, if $(\alpha, \beta) \in P^{\mathcal{I}}$ for some role name P then
   - both α and β correspond to individual names;
   - or β is equal to α concatenated to an element of the alphabet domain;
   - or α is a subsequence of β and there is a transitive role S included in P connecting all the subsequences of β having α as subsequence (i.e. all the sequences between α and β).
2. Given two arbitrary sequences α, β in $\Delta^{\mathcal{I}}$, if the pair $(\alpha, \beta) \in P^{\mathcal{I}} \cap R^{\mathcal{I}}$ for some different role names P, R then
   - both α and β correspond to individual names;
   - or the role P is included in R (or vice versa);
   - or there are functional roles $F_1, \ldots, F_n$ and roles $R_0, R_1, \ldots, R_n$ s.t. $P = R_0$ and $R = R_n$, both R_{i-1} and R_i are included in F_i for $i = 1, \ldots, n$, and $(\alpha, \beta) \in \bigcap_{i=0,\ldots,n}(R_i)^{\mathcal{I}}$.

Given the properties of q.t. shrubs, we cannot assume that variables appearing in cyclical queries must be identified with variable names. For simple cycles we must verify the two alternative cases as well (the roles are included one into the other, or there is a chain of functional roles with the properties showed above). For complex cycles we must verify whether part of a cycle can be satisfied by transitive roles.

5 Conclusions

In this paper we show how the query answering technique presented in [7] can be extended to DLs featuring complex restrictions on the interpretation of roles (i.e. transitivity, role hierarchy, and functionality).

Note that the DL we are considering takes a different approach w.r.t. the logic PDL used in [1] and [2].[8] In PDL there are constructs for expressing complex role expressions using atomic role names (e.g. role disjunction and transitive closure), but they do not affect the tree–model property of PDL which is essentially the same as for $\mathcal{ALC}$ (as described in Definition 1).

Investigating query answering for DLs like $\mathcal{SH}f$ is relevant because most of the available state of the art DL systems (see [4, 5]) implement DLs with transitive roles and role hierarchy instead of transitive closure of roles and/or role disjunction (or conjunction). Therefore, results for PDL and Converse PDL cannot be easily used for these systems.

In devising suitable query answering algorithms, using the technique described in this paper, we are facing two distinct problems. The first consist in identifying the right class of interpretations and showing its completeness w.r.t. the problem of query answering. The second is designing the collapsing rules using the restrictions on interpretations belonging to the identified class.

We are currently working on the problem of extending the framework to the logic $\mathcal{SHIQ}$, which provides inverse role and qualified number restriction operators on top of $\mathcal{SH}f$.

References

[1] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the Symposium on Principles of Database Systems (PODS-98)*, pages 149–158, 1998.

[2] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[8] In fact, this approach uses Converse PDL with qualified number restrictions, but KB satisfiability is reduced to the formula satisfiability problem in PDL (see [3]).

[3] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann Publishers, 1996.

[4] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In *Proc. of TABLEAUX 2000*, volume 1847 of *Lecture Notes in Computer Science*, pages 57–61. Springer, 2000.

[5] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[6] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Logic for Programming and Automated Reasoning (LPAR 2000)*, volume 1955 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2000.

[7] I. Horrocks and S. Tessaris. Answering conjunctive queries over DL Aboxes: A preliminary report. In *Proc. of the 2000 International Workshop on Description Logics (DL2000)*, pages 173–182, 2000.

[8] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.

[9] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic.* PhD thesis, University of Manchester, 2001.

[10] S. Tessaris and G. Gough. Abox reasoning with transitive roles and axioms. In *Proc. of the 1999 International Workshop on Description Logics (DL'99)*, pages 101–104, 1999.

[11] M. Y. Vardi. Why is modal logic so robustly decidable? Technical Report TR97-274, Rice University, 1997.