Scalability of OWL Reasoning: Role condensates

Sebastian Wandelt[1], Ralf Möller[1]

Institute for Software Systems,
TU Hamburg-Harburg,
wandelt@tuhh.de, r.f.moeller@tuhh.de

Abstract. In the last years, there has been an increasing interest in the performance of reasoning on the Semantic Web in presence of large ABoxes. Traditional reasoners make heavily use of in-memory structures and are therefore not suitable to deal with large ABoxes directly.
We propose an approach that, informally speaking, works on a composite representation of the role-part of a SHIQ knowledge base. With respect to conjunctive queries, this helps us to provide a kind of proxy that restricts the set of possible bindings for a variable in advance. Furthermore we can use this proxy to reject several queries with no answer substitution immediately. Most notably our approach is query independent.

1 Introduction

As the Semantic Web evolves, scalability of inference techniques becomes increasingly important. Even for basic inference techniques, e.g. concept satisfiability, it is only recently understood on how to perform reasoning on huge input in an efficient way. This is not yet the case for problems that are too large to fit into main memory. For more complex reasoning problems like conjunctive query answering, this problem becomes even harder.

It has been shown recently [GHLS07] that answering conjunctive queries over a SHIQ knowledge base is decidable. The data complexity (complexity w.r.t. the size of the ABox) was shown to be co-NP complete. Given that result, dealing with large ABoxes (10^6 and more assertions) is likely to be unfeasible in practice for the general/worst case.

In this paper we present a complete, but unsound approach to answer conjunctive queries in a proxy-like manner. For our approach we have a look at the role part of a knowledge base. For a given SHIQ knowledge base, we create a condensed role graph. Usually, this graph is several orders of magnitudes smaller than the ABox. Furthermore, we convert a conjunctive query into a number of forest-like structures. Then we show that the conjunctive query is only entailed by the knowledge base if one query forest is isomorphic to a subgraph of the condensed graph.

One might claim that the subgraph isomorphism problem is NP-complete and that we have not much gain for the price of unsoundness. Yet, we advocate, that

our data set is smaller. Thus, the impact of the high worst-case complexity is not so dramatic, and the structures fit into main memory. Furthermore, subgraph isomorphism checking is a very fundamental problem of computer science and well-known heuristics can be applied to speed up the algorithm in practice.

This paper is structured as follows. Section 2 presents some background on query answering over SHIQ ontologies. In Section 3 we introduce the notions of a role condensate and in Section 4 we show an appropriate role-encoding of conjunctive queries. Section 5 provides our main reasoning result. We conclude the paper in Section 6 and also point at some ideas for further work.

2 Basics

First, we briefly introduce the syntax and semantics of the description logic SHIQ. We assume a collection of disjoint sets: a set of *concept names* N_C , a set of *role names* N_{RN}, with a subset $N_{RN+} \subseteq N_{RN}$ of transitive role names, and a set of *individual names* N_I. The *set of roles* N_R is $N_{RN} \cup \{R^-|R \in N_{RN}\}$. We also define the following functions for inverse and transitive roles:

$$\begin{aligned} Inv(R) = R^- &\iff R \in N_{RN} \\ Inv(R) = S &\iff R = S^- \\ Trans(R) &\iff R \in N_{RN+} \vee Inv(R) \in N_{RN+} \end{aligned}$$

SHIQ-concepts are built inductively by using the grammar:

$$C ::= \top|\bot|D|\neg C|C_1 \sqcap C_2|C_1 \sqcup C_2|\forall R.C|\exists R.C|\leq_n S.C|\geq_n S.C,$$

where $D \in N_C$, $R \in N_R$, $S \in N_R$
N_{RN+} and $n \in \mathbb{N}$.

We define the semantics by using a standard Tarski-style semantics with an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \bullet^{\mathcal{I}})$, consisting of a non-empty set (*domain*) $\Delta^{\mathcal{I}}$ and a function (*valuation*) $\bullet^{\mathcal{I}}$ which maps every concept name D to a subset $D^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, every role name $R \in N_R$ to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, while satisfying the following equations:

$$\begin{aligned} (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\ (\exists R.C)^{\mathcal{I}} = \{x|\exists y.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}, \\ (\forall R.C)^{\mathcal{I}} = \{x|\forall y.(x,y) \in R^{\mathcal{I}} \implies y \in C^{\mathcal{I}}\} \\ (\geq_n R.C)^{\mathcal{I}} = \{x|\,|\{y|(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\}, \\ (\leq_n R.C)^{\mathcal{I}} = \{x|\,|\{y|(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leq n\}, \end{aligned}$$

A *general concept inclusion* is an expression $C \sqsubseteq D$, where C and D are concepts. A finite set of general concept inclusions is called *TBox*. A *role inclusion* is of the form $R \sqsubseteq S$ with R and S roles. A *role hierarchy* is a finite set of role

inclusions. With $\sqsubseteq^*$ we denote the transitive closure of $\sqsubseteq$. An *assertion* is of the form $C(a), \neg C(a), R(a,b), \neg R(a,b)$ or $a \neq b$, where C is a concept, R is a role and $a, b \in N_I$. A finite set of assertions is called *ABox*. With $Ind(\mathcal{A})$ we denote the set of individuals occurring in $\mathcal{A}$. A *knowledge base* KB is a triple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ that consists of a TBox $\mathcal{T}$, a role hierarchy $\mathcal{R}$ and an ABox $\mathcal{A}$.

An interpretation $\mathcal{I}$ is a *model* of a concept C, if $C^{\mathcal{I}} \neq \emptyset$. An interpretation $\mathcal{I}$ is a model of a general concept inclusion $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation is a model of a TBox T if it satisfies all axioms in T. Being model of an RBox and ABox is defined as usual. An interpretation is a model of a knowledge base, if it is a model of $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

In the following we use a common definition of conjunctive queries, as e.g. given in [GHLS07]. Let N_V be a countably infinite set of variables. A *concept atom* is an expression $C(v_1)$ and a *role atom* is an expression $R(v_1, v_2)$, where C is a concept name, R is a role, and $v_1, v_2 \in N_V$. A *conjunctive query* Q is a non-empty set of atoms. Let $Var(Q)$ denote the set of variables in a query Q.

We write for an interpretation $\mathcal{I}$, a conjunctive query Q and a total function $\pi : Var(Q) \to \Delta^{\mathcal{I}}$:

1. $\mathcal{I} \models^{\pi} C(v)$ if $(\pi(v)) \in C^{\mathcal{I}}$
2. $\mathcal{I} \models^{\pi} R(v_1, v_2)$ if $(\pi(v_1), \pi(v_2)) \in R^{\mathcal{I}}$

Furthermore, we write $\mathcal{I} \models^{\pi} Q$ if we have $\mathcal{I} \models^{\pi} A$ for all atoms A in Q. If $\mathcal{I} \models^{\pi} Q$ for all models $\mathcal{I}$ of a knowledge base KB and some π, we write $KB \models Q$.

Given a knowledge base KB and a query Q, the *query entailment problem* is to decide whether $KB \models Q$. It is folklore, that query entailment and query answering problems are equivalent [HT00].

A conjunctive query Q is *connected* if, for all $v, v' \in Var(Q)$, there exists a sequence $v_0, ..., v_n$ such that $v_0 = v$ and $v_n = v'$ and for all $i < n$, there exists a role R, s.t. $R(v_i, v_{i+1}) \in Q$.

A *role hierarchy graph* is a directed graph $\mathcal{G}_{\mathcal{H}} = \langle N, E \rangle$, where roles are the nodes and there is a directed edge from R to S, whenever $R \sqsubseteq S$. We assume that the graph is reflexive, that is, each node has an edge directed to itself. Usually, a role hierarchy graph will contain several unconnected subgraphs. We call two roles S, T *hierarchically connected*, written $S \cong T$, if they belong to the same subgraph.

With $[R]_H$ we denote the equivalence class (the subgraph of $\mathcal{G}_{\mathcal{H}}$) R is contained in. That is, $R \cong S \iff [R]_H = [S]_H$. We extend our definition of $Trans$ in such a way, that $Trans([R]_H) \iff (\exists S \in [R]_H . Trans(S))$. Let $N_{[R]_H} = \{[R]_H | R \in N_R\}$.

3 Determining role condensates

Let $KB = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ be a SHIQ knowledge base. We want to build a graph G from KB, that has the following property: Whenever $KB \models R_1(i_1, i_2) \wedge R_2(i_2, i_3) \wedge ... \wedge R_{n-1}(i_{n-1}, i_n)$, for named or fresh (=not occuring in the ABox)

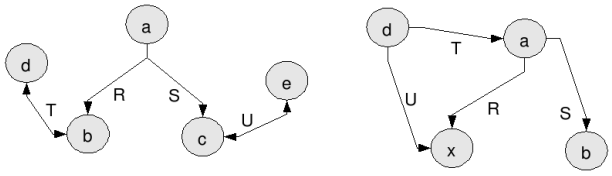


Fig. 1. Possible merging of individuals

individuals $i_1 ... i_n$, then we have a path $f(i_1) \rightarrow_{[R_1]_H} ... \rightarrow_{[R_n]_H} f(i_n)$ in G, where f is a homomorphism and $\rightarrow_X$ stands for a directed X-edge.

The intuition is to find a graph, s.t. anything that can be observed about role-transitions in a knowledge base, can also be observed about role transitions in our graph, but not vice versa. We create this graph G in two separate steps:

1. We create a small graph, that creates a superset of possible roles between named individuals.
2. We extend this graph by applying a kind of worst-case algorithm to derive further information about all possible roles that can be inferred by $\exists R$- and $\geq_n R$-constraints

Let us define the graph structure for the first step:

Definition : A *role condensate* (of a knowledge base KB) is a graph $\mathcal{G}_\mathcal{A} = \langle V, E, \phi, \omega \rangle$, where $V \subseteq \mathbb{N}$, $E \subseteq V \times V \times 2^{N_{[R]_H}}$, $\phi : V \rightarrow 2^{Ind(\mathcal{A})}$ and $\omega : V \rightarrow 2^{clos(KB)}$.

In the rest of this paper we will often use the inverse of ϕ. This is a function as well, since we enforce $\forall x, y \in V . \phi(x) \cap \phi(y) = \emptyset$. Whenever we call a role condensate *grounded*, we assume, that ω is empty.

To build the graph, notice that in a SHIQ-knowledge base one never removes existing roles between two named individuals. The known algorithms only infer additional roles between named individuals based on given constraints. These constrains are transitivity, role hierarchies and maximum cardinality constraints. Let us have a look at these three cases:

- **Transitivity**:
  Whenever we have $KB \models R(a, b)$, $KB \models R(b, c)$ and $Trans(R)$, then we can conclude, that $KB \models R(a, c)$.
- **Role hierarchies**:
  Whenever we have $KB \models S(a, b)$ and $S \sqsubseteq^* R$, then we can conclude, that $KB \models R(a, b)$.
- **Maximum cardinality constraints**:
  In the following we will discuss the case of two mergable nodes. This can be easily extended for the case of n nodes. There are two distinct situations to consider.

1. We have to merge two named individuals b and c (Figure 1, left). This can only happen, if $a :\leq_n V.X$ and $R \cong V \cong S$. The merged individuals will have all the neighbors of b plus all the neighbors of c. Thus, if we replace all roles R by $[R]_H$, then we can merge named individuals based on role equality.
2. We have to merge a named individuals b and an unnamed individual x (Figure 1, right). The unnamed individual x can only be connected to another (arbitrary) named individual d, if $T \cong U \cong R$ and $Trans([U]_H)$. Furthermore, x and b will only be merged if $R \cong S$. Thus, if we replace all roles R by $[R]_H$, then we can simulate named-unnamed merging by explicitly closing the graph for transitivity.

The above facts tell us what we have to do to create a superset of possible roles between named individuals for our role condensate $\mathcal{G}_\mathcal{A}$. Yet, we also want to reduce the number of nodes in $\mathcal{G}_\mathcal{A}$. We want to merge similar individuals if they are connected to the same individual via a role $[R]_H$. The key is to have an adequate measure of equality of 2 (or n) named individuals. Let us define an abstract *individual similarity relation* $Sim(a, b) : Ind(\mathcal{A}) \times Ind(\mathcal{A})$. Three possible instances of this relation (among others) are:

1. **Same set of in-/outgoing roles:** We could define $Sim(a, b) \iff (S_{Role}(a) = S_{Role}(b))$, where $S_{Role}(a)$ is the union of incoming and outgoing roles of i.
2. **Same concept sets:** We could define $Sim(a, b) \iff (S_{Conc}(a) = S_{Conc}(b))$, where $S_{Conc}(a)$ is the set of concepts associated to an individual/node a.
3. **All individuals are similar:** We could define $Sim(a, b) = Ind(\mathcal{A}) \times Ind(\mathcal{A})$.

In the following we will use Sim without making a commitment to particular instances. Next, we propose a way to create a role condensate from a given SHIQ KB. The algorithm is shown in figure 2.

Input: SHIQ knowledge base $KB = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$
Output: Grounded role condensate $\mathcal{G}_\mathcal{A}$
Algorithm:

1. **For each** $R(a, b) \in \mathcal{A}$ **do**
   (a) $v_a = getOrCreate(\mathcal{A}, \mathcal{G}_\mathcal{A}, a)$
   (b) $v_b = getOrCreate(\mathcal{A}, \mathcal{G}_\mathcal{A}, b)$
   (c) **Add** $[R]_H(v_a, v_b)$ to $\mathcal{G}_\mathcal{A}$ and **add** $Inv([R]_H)(v_b, v_a)$ to $\mathcal{G}_\mathcal{A}$
   (d) **While** $\mathcal{G}_\mathcal{A}$ is changed below
      i. **If** $(\exists c, d, e.S(c, d) \in \mathcal{G}_\mathcal{A} \wedge S(c, e) \in \mathcal{G}_\mathcal{A} \wedge d \neq e \wedge (Sim(d, e) \vee \leq_n S.X \in clos(KB))$ **then** $Merge(\mathcal{G}_\mathcal{A}, d, e)$
      ii. **If** $(\exists c, d, e.S(c, d) \in \mathcal{G}_\mathcal{A} \wedge S(d, e) \in \mathcal{G}_\mathcal{A} \wedge S(c, e) \notin \mathcal{G}_\mathcal{A} \wedge Trans([S]_H))$ **then add** $[S]_H(c, e)$ to $\mathcal{G}_\mathcal{A}$

Fig. 2. Grounded role condensates: algorithm

The algorithm should be self-explaining. We will only have a look at the while-loop of the main algorithm.

Function *merge*
Parameter: $\mathcal{G}_\mathcal{A}$, nodes d, e
Algorithm:
1. **For each** $R(x, e) \in \mathcal{G}_\mathcal{A}$ **do**
   (a) **Add** $R(x, d)$ to $\mathcal{G}_\mathcal{A}$
   (b) **Remove** $R(x, e)$ from $\mathcal{G}_\mathcal{A}$
2. **For each** $R(e, y) \in \mathcal{G}_\mathcal{A}$ **do**
   (a) **Add** $R(d, y)$ to $\mathcal{G}_\mathcal{A}$
   (b) **Remove** $R(e, y)$ from $\mathcal{G}_\mathcal{A}$
3. **Set** $\phi(d) = \phi(d) \cup \phi(e)$
4. **Set** $\omega(d) = \omega(d) \cup \omega(e)$
5. **Remove** e from $\mathcal{G}_\mathcal{A}$

Function *getOrCreate*
Parameter: $\mathcal{A}$, $\mathcal{G}_\mathcal{A}$, individual a
Returns: node n
Algorithm:
1. **If** $a \notin \bigcup_{v \in V} \phi(v)$ **then**
   (a) **Add** new node n to $\mathcal{G}_\mathcal{A}$
   (b) **Set** $\omega(n) = \{C | a : C \in \mathcal{A}\}$
   (c) **Set** $\phi(n) = \{a\}$
   (d) **Return** n
2. **else**
   (a) **Return** v, s.t. $\phi(v) = a$

Fig. 3. Grounded role condensates: helper functions

The first part of the loop merges individuals if they are similar or might have to be merged due to a maximum cardinality restriction. The latter is determined in a rather coarse-grained approach. We only look at the closure of all concepts in the knowledge base. We could do better by applying a similar strategy as in [FKM$^+$06]. They carefully analyze the ABox and determine to which individuals maximum cardinality restrictions can be forwarded.

The second part of the while-loop creates all possible transitive edges between individuals.

Lemma (*homomorphic monotonicity*): Let $\mathcal{G}_\mathcal{A}$ be the role condensate of $\mathcal{A}$. Whenever the algorithm adds to $\mathcal{G}_\mathcal{A}$ an R-edge from $\phi^-(a)$ to $\phi^-(b)$, then $\mathcal{G}_\mathcal{A}$ will also have an R-edge from $\phi^-(a)$ to $\phi^-(b)$ after the algorithm terminates.

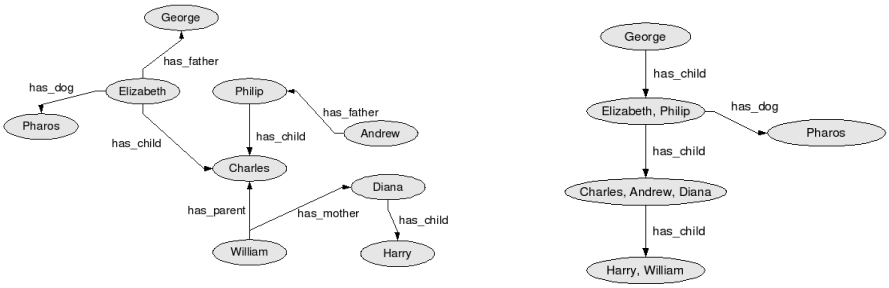
Proof: Can be done by case analysis of the algorithm. □

Theorem 1. *Let $\mathcal{G}_\mathcal{A}$ be the grounded role condensate of $\mathcal{A}$. Whenever we have $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \models R(a, b)$, for two named individuals a and b, then we have that $[R]_H(\phi^-(a), \phi^-(b)) \in \mathcal{G}_\mathcal{A}$.*

Proof: This can be done by induction. The base case is that the theorem is true for all existing role relationships in the source ABox. The induction steps are based on possible reasons for adding an R-edge between two named individuals and the use of homomorphic monotonicity lemma. □

We emphasize that the opposite direction of this theorem is not true. We do have complete, but unsound, reasoning on the role-relationships between named individuals in the ABox. The above result can be easily lifted to the case of paths between named individuals.

To make our approach clear, we will demonstrate it with a simple example. Let us consider the 'messed up' snapshot of the british royal family shown in Figure 4(a). We assume the following role information: $has\_father \sqsubseteq has\_parent$, $has\_mother \sqsubseteq has\_parent$ and $Inv(has\_parent) = has\_child$. The result of the



(a) Source ABox (b) Result

Fig. 4. Example 1: Grounded role condensates

algorithm is shown in 4(b) (inverse roles are not explicitly drawn). Please note that in the grounded role condensate all individuals are grouped w.r.t. their level in the family's genealogical tree.

With the grounded role condensates we have a kind of base skeleton for the distribution of roles in a SHIQ knowledge base. Additional roles can only exist between named and unnamed individuals, and also between two unnamed individuals. These role relationships can only be created by concept expressions such as $\exists R.X$ and $\geq_n R.X$. We call these two expressions *R-generators* from now on.

To extend the role condensates to the non-grounded case, we apply a kind of worst-case tableaux algorithm to a grounded role condensate $\mathcal{G}_\mathcal{A}$. Note that after running the tableau algorithm, we set blocked nodes equal to their respective blockers, to capture infinite paths in the tree.

First, we initialize ω of $\mathcal{G}_\mathcal{A}$ as follows: $\forall v \in V.(\omega(v) = \{C | \exists a \in \phi(v).C(a) \in \mathcal{A}\})$. The tableau-like algorithm is shown in figure 5.

$\sqcap$-rule	If $C_1 \sqcap C_2 \in \omega(x)$, x is not indirectly blocked and $\{C_1, C_2\} \nsubseteq \omega(x)$, then $\omega(x) = \omega(x) \cup \{C_1, C_2\}$
$\sqcup$-rule	If $C_1 \sqcup C_2 \in \omega(x)$, x is not indirectly blocked and $\{C_1, C_2\} \nsubseteq \omega(x)$, then $\omega(x) = \omega(x) \cup \{C_1, C_2\}$
$\exists$-rule	If $\exists R.C \in \omega(x)$, x is not blocked and x has no $[R]_H$-neighbor y with $C \in \omega(y)$, then create a new node y with $\omega(y) = \{C\}$ and $E = E \cup \{(x, y, [R]_H)\}$
$\forall$-rule	If $\forall R.C \in \omega(x)(x)$, x is not indirectly blocked and there is an $[R]_H$-neighbor of y of x with $C \notin \omega(x)(y)$, then $\omega(y) = \omega(y) \cup \{C\}$
$\geq_n$-rule	If $\geq_n R.C \in \omega(x)(x)$, x is not blocked and x has no $[R]_H$-neighbor y with $C \in \omega(x)(y)$, then create a new node y with $\omega(y) = \{C\}$ and $E = E \cup \{(x, y, [R]_H)\}$
$\forall_+$-rule	If $\forall R.C \in \omega(x)$, x is not indirectly blocked, we have $Trans(R)$ and there is an R-neighbor y of x, with $\forall R.C \notin \omega(y)$, then $\omega(y) = \omega(y) \cup \{\forall R.C\}$

Fig. 5. Worst-case role tableau algorithm

Theorem 2. *Let $KB = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ be a SHIQ knowledge base and let $\mathcal{G}_{\mathcal{A}}$ be the role condensate of $\mathcal{A}$. Whenever $KB \models R_1(i_1, i_2) \wedge R_2(i_2, i_3) \wedge ... \wedge R_{n-1}(i_{n-1}, i_n)$, for named or fresh individuals $i_1 ... i_n$, then we have a path $\phi^-(i_1) \rightarrow_{[R_1]_H} ... \rightarrow_{[R_n]_H} \phi^-(i_n)$ in $\mathcal{G}_{\mathcal{A}}$.*

Proof: Is a direct consequence of Theorem 1 and of the exhaustive application of worst-case tableaux rules. □

Notice, that the opposite direction is again not true here. That is, the role condensate might create new subgraphs due to role condensation. Now we have a graph encoding of the roles between individuals in a knowledge base, the role condensate $\mathcal{G}_{\mathcal{A}}$. In the next section we show how to transform conjunctive queries into a set of forests that can be checked for being a subgraph of $\mathcal{G}_{\mathcal{A}}$.

4 Representing role-parts of conjunctive queries

This section shows how to use role condensates to answer conjunctive queries in an unsound, but complete manner. Without loss of generality we assume that the conjunctive query Q is connected. Otherwise we split Q into n connected queries q_i and answer each of them separately.

Definition : A *query forest match* (of a conjunctive query Q) is a forest $\mathcal{G}_{\mathcal{Q}} = \langle V, E \rangle$, where $V \subseteq Var(Q) \cup \mathbb{N}$ and $E \subseteq V \times V \times N_R$. A query forest match $\mathcal{G}_{\mathcal{Q}}$ is created non-deterministically by the algorithm shown in figure 6.

The idea of a query forest match is to encode one possible forest structure (role-wise) which a conjunctive query can enforce. In contrast to *query graphs* [GHLS07], we do not (in general) only regard role query atoms, but also possible unfoldings of the concept query atoms. This is done by the helper function *conceptPathway*. It creates one tree representing role paths for each possible (non-deterministic) unfolding. The non-determinism is introduced by the disjunction of concepts. The function uses three not further explained helper functions to create trees: *mergeRoots* (merges two sibblings), *leaf* (creates a leaf of a tree) and *newRoot* (creates a new root node for a subtree).

Function: $getQueryForestMatch$
Input: Knowledge base KB, Conjunctive query Q, Max depth d
Output: Query forest match $\mathcal{G}_{\mathcal{Q}}$
Algorithm:
1. **Let** $\mathcal{G}_{\mathcal{Q}}$ be the graph created by the role query atoms in Q
2. **For each** $C(X) \in Q$ do
   (a) **Add** one t, s.t. $t \in conceptPathway(\mathcal{KB}, C, d)$, to node X in $\mathcal{G}_{\mathcal{Q}}$

Fig. 6. Query forest match algorithm

Function: $conceptPathway$
Input: Knowledge base KB, Concept C, Max depth d
Output: Tree T

1. **If** $d = 0$ **then**
   return $T = leaf()$
2. **If** $C = C_1 \sqcap C_2$ **then**
   return $T = mergeRoots(conceptPathway(C_1, d), conceptPathway(C_2, d))$
3. **If** $C = C_1 \sqcup C_2$ **then**
   return $T = conceptPathway(C_n, d-1)$, for one $n \in \{1, 2\}$
4. **If** $C = \exists R.C_1$ or $C = \geq_n .C_1$ **then**
   return $T = newRoot(R, conceptPathway(C_1))$
5. **If** $(C \sqsubseteq C_1) \in \mathcal{T}$ **then**
   return $T = conceptPathway(C_1, d-1))$
6. **else return** $T = leaf()$

Fig. 7. Query forest match algorithm - *conceptPathway* function

Notice that we restrict the unfolding to a particular depth d. First, since we might have cyclic GCIs, we guarantee termination that way. Second, when we combine the set of unfoldings for each concept of a named individual, we obtain a possibly exponential amount of forests. This is not desirable for practical reasoning. Still, we want to emphasize that our approach is complete for each $d \in \mathbb{N}$. For $d = 0$ we obtain the standard query graph.

5 Answering conjunctive queries

Let Q_F be the set of all possible query forest matches for a query Q, that is, $Q_F = \{\mathcal{G}_\mathcal{Q} | getQueryForestMatch(KB, Q, d)\}$.

Theorem 3. *Let $KB = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ be a SHIQ knowledge base and Q be a conjunctive query. If we have that $KB \models Q$, then we can find a query forest match $\mathcal{G}_\mathcal{Q}$, that is isomorphic to a subgraph of $\mathcal{G}_\mathcal{A}$.*

Proof: The proof follows directly from Theorem 2. □

We applied our approach to the LUBM Ontology [GPH05], an ontology benchmark in the setting of universities. It is accompanied by a set of 14 test queries. The second query of that set is interesting for our approach, since it has no solution for most data generations of the LUBM Generator. This query is shown in Figure 8, using a SPARQL-like syntax.

```
(type GraduateStudent ?X), (type University ?Y), (type Department ?Z),
(memberOf ?X ?Z), (subOrganizationOf ?Z ?Y), (undergraduateDegreeFrom ?X ?Y)
```

Fig. 8. LUBM query 2

First, we have created a role condensate for several LUBM(X,0) (where X is the number of universities). The result is shown in figure 9. For the individual similarity relation Sim, we have used the one with 'same set of incoming/outgoing roles'.

Universities	Individuals	Sim_1:nodes in $\mathcal{G}_\mathcal{A}$
5	102K	25
10	205K	24
20	407K	26

Fig. 9. LUBM statistics

Let Q be the conjunctive query shown in Figure 8. Setting the maximum depth for the concept unfolding $d = 0$, then we obtain a single query forest match $\mathcal{G}_\mathcal{Q}$, that corresponds to the query graph of Q. When we check whether $\mathcal{G}_\mathcal{Q}$ is isomorphic to a subgraph of $\mathcal{G}_\mathcal{A}$, we get a *no* immediately, for all tested number of universities. This small example shows already that our approach *can* dramatically speed up the rejection of queries with no answers.

6 Conclusion and Future Work

The decision procedure presented in this work makes it possible to answer conjunctive queries in a complete, but unsound, way. We obtain a proxy-like decision system which can possibly immediately reject conjunctive queries having no answer replacements. Moreover, it can be used to provide a set of obvious non instances for each variable in the grounded setting (e.g. the query language NRQL [HMW04]). This will be discussed in future work.

Until now, our approach works only for the description logic SHIQ. For OWL DL, that is SHOIQ, we need to further add handling of nominals. This is part of future work. We will also investigate an upper bound for the complexity of role condensate creation. Furthermore, we will apply the approach to additional test ontologies (especially non-LUBM) in order to provide detailed statistics about its usefulness in practice.

References

[FKM+06] Achille Fokoue, Aaron Kershenbaum, Li Ma, Chintan Patel, Edith Schonberg, and Kavitha Srinivas. Using Abstract Evaluation in ABox Reasoning. In *SSWS 2006*, pages 61–74, Athens, GA, USA, November 2006.

[GHLS07] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic shiq. In *IJCAI-07*, 2007.

[GPH05] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[HMW04] V. Haarslev, R. Möller, and M. Wessel. Querying the semantic web with racer + nrql. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24*, 2004.

[HT00] Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.