FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN - BOLZANO

KRDB RESEARCH CENTRE
KNOWLEDGE REPRESENTATION MEETS DATABASES

Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009, http://www.inf.unibz.it/krdb/

KRDB Research Centre Technical Report:

Extended theoretical foundations for QUELO framework

Nhung Ngo

Affiliation	KRDB Research Center, FUB
Corresponding author	Ngo Thi Phuong Nhung maxnhong85@gmail.com
Keywords	query building, ontology based data access, relational database
Number	KRDB10-2
Date	05-05-2010
URL	http://www.inf.unibz.it/krdb/

Abstract

QUELO framework [Catarci, 2004] [Dongilli,2004] has been built by a group led by Professor Franconi since several years ago and it's foundations were proposed by Guagliardo Paolo in his master's thesis [Paolo,2009]. The aim of this project is to extend Paolo's works by considering queries with constants, attributes and queries over relation databases instead of ABoxes. Regarding to constants and attributes, we will show that these elements will not affect our reasoning services in $\mathcal{ALC}$. In order to answer queries over relation databases, a set of mapping rules between relational databases constraints and DL Knowledge bases will be introduced. We also will prove that the query refinement process in our framework works normally with these mapping rules.

Contents

1 Introduction

Nowadays, people more and more concern about using ontologies to represent information systems and there are many techniques supporting querying over ontologies. In general, ontologies are huge then users don't know exactly their content. As a consequence, when they want to query some information, they don't know how to express it. That's why a tool to help users building their queries is necessary.
From this motivation, the first version of a tool named Query Tool was developed under the SEmantic Webs and AgentS in Integrated Economies (SEWASIE) research project [Catarci, 2004]. With this tool, users can build a query over an ontology from the scratch and refine it. The refinement process makes sure that the query is consistent with respect to the ontology. The main features [Paolo,2009] of this tool are following:

- Visual interface : This visual interface allows users manipulate the query
- Focus paradigm: Users can refine a query by select a subquery. All the refine options such as : getting more general one or more specific one are based on this selected subquery named *focus*.
- Refinement by compatible term: A focus can be replaced by only its compatible terms. These terms are collected by using some Description Logic reasoning such as Pellet or Fact++.

In order to explain the theoretical foundations of the tool above as well as its operations, a formal Query Tool framework was proposed [Paolo,2009]. In this formal framework, a conjunctive query and its refinements are defined in an unambiguous way such as :

- A conjunctive query is define as a tree and a focus term is corresponding to a subtree. A query with a focus term can be transformed to a Description logic concept by a rolling up procedure.
- All the query refinement operations like: Add compatible, Add relation, Select, Delete, Weaken and Substitute are written as functional APIs which can support different type of interfaces.

As we mentioned in the abstract, in the above framework, they did not concern about the constants and datatypes, which are all supported by expressive description logics. Therefore, in the first part of this project, we will extend it by consider queries with constants and datatypes. In the second part, we will consider this framework together with relation databases by giving a mapping between a relation database and a description logic knowledge base.

2 Queries with constants and attributes

In this section, we will consider queries which contain constants or attributes such as :

- $Q(x) = Person(x) \wedge Married(x, Mary) \wedge Person(Mary) \wedge ...$
- $Q(x) = Person(x) \wedge hasName(x, \text{“}John\text{”}) \wedge String(\text{“}John\text{”}) \wedge ...$

We will show that actually, our system can treat these queries as $Q(x) = Person(x) \wedge \exists y.(Married(x, y) \wedge Person(y)) \wedge ...$ or $Q(x) = Person(x) \wedge \exists y.hasName(x, y) \wedge ..$ without changing their answers in $\mathcal{ALC}$

2.1 Constants

Theorem 1 *$KB \models_{FOL} \top \sqsubseteq Q(x, a)$ iff $KB \models_{FOL} \top \sqsubseteq Q(x, b)$ for arbitrary a,b where :*

- *KB is a $\mathcal{ALC}$ KB*
- *$Q(x, y) \equiv \exists z.T(x, z) \wedge C(z) \wedge y = z$ and C is a concept name.*

Proof:
Since the roles of a and b are similar, we just need to prove 1 direction. Consider $KB \models_{FOL} \top \sqsubseteq Q(x,a)$*, we have to prove* $KB \models_{FOL} \top \sqsubseteq Q(x,b)$
Call I *is an arbitrary model of* KB*, by the assumption then* $\forall x \in \Delta^I, (x, a^I) \in T^I$*. Assume that* I *is not a model of* $\top \sqsubseteq Q(x,b)$*, then there is* $x' \in \Delta^I$ *such that* $(x', b^I) \notin T^I$
Call I' *is an interpretation such that* $\Delta^{I'} = \Delta^I$*,* $\cdot^{I'}$ *is the same with* $\cdot^I$ *except* $a^{I'} = b^I$ *and* $b^{I'} = a^I$*. Since the* KB *is in* ALC *then there is no fact (axioms and assertions) containing* a *or* b*. Therefore,* I' *is also a model of* KB*.*
However, by the definition of I' *we have : there is a* $x' \in \Delta^{I'}$ *such that* $(x', a^{I'}) \notin T^{I'}$*. Then* I' *is not a model of* $\top \sqsubseteq Q(x,a)$
$\Rightarrow$ *Contradiction*
$\Rightarrow$ I *is also a model of* $\top \sqsubseteq Q(x,b)$
Therefore, $KB \models_{FOL} \top \sqsubseteq Q(x,b)$

Based on above theorem, it's clear for us to treat the query $Q(x) \equiv \exists z.T(x,z) \wedge C(z) \wedge a = z$ (where a is a constant) as a query $Q(x) \equiv \exists z.T(x,z) \wedge C(z)$ because the constant a will not play any role in the answer of this query. The rewritten query will be process normally in our framework

2.2 Attributes

In terms of attributes, the analogous theorem will hold.

Theorem 2 $KB \models_{FOL} \top \sqsubseteq Q(x,a)$ *iff* $KB \models_{FOL} \top \sqsubseteq Q(x,b)$ *for arbitrary a,b where*

- KB *is a* $\mathcal{ALC}$ KB
- $Q(x,y) \equiv \exists z.T(x,z) \wedge D_v(z) \wedge y = z$ *and* D_v *is a datatype*

As a consequence, we can rewrite the query $Q(x) \equiv \exists z.T(x,z) \wedge D_v(z) \wedge z = a$ as $Q(x) \equiv \exists z.T(x,z) \wedge D_v(z)$. However, attributes should be consider in the framework in a slightly different way by mean that they will occur only as leaves of queries. In the other words, we can not add relations, add attributes or role up at their positions.
Therefore, in the remain part of this section, we will modified some formal definitions, procedures for functional API and the algorithms to support these changes.

a) Formal definitions

Definition 3 (Query) *Let* $\boldsymbol{N}$ *be a countable set of node names,* $\boldsymbol{C}$ *a finite set of concept names,* $\boldsymbol{R}$ *is a finite set of role names,* $\boldsymbol{A}$ *is a finite set of attribute names and* $\boldsymbol{D}$ *is a finite set of datatypes and* $\boldsymbol{N, C, R, A, D}$ *be pairwise disjoint. A query* Q *is a quintuple* $\langle V, E, o, \mathcal{V}, \mathcal{E} \rangle$ *where :*

- (V, E) *i a directed tree rooted in* $o \in V$*, in which* $V \subset N$ *is the set of nodes and* $E \subset V \times V$ *is the set of (directed) edges.*
- $\mathcal{V} : V \to (2^C \backslash \{\emptyset\}) \cup \{\{\tau\}\} \cup \{\{d | d \in D\}\}$ *is a total function, called node-labelling function, which associates each node with a non-empty set of concept name or a datatype.*
- $\mathcal{E} : E \to R \cup A$ *is a total function, called edge-labelling function, associating each edge with a role name or an attribute name.*

such as :

- *For all* $(v, v') \in E$*, there is no* $d \in D$ *s.t* $\mathcal{V}(v) = \{d\}$
- *For all* $v \in V$ *such that* $\mathcal{V}(v) = \{d\}$ *and* $d \in D$ *:*

– *There is no v' such that $(v, v') \in E$*
– *There is a v'' such that $(v'', v) \in E$ and $\mathcal{E}(v'', v) \in A$*
– *v is called **datatype node***

Definition 4 (Subquery) *Given queries S and Q, we say that S is a subquery of Q, and write $S \subseteq Q$ iff all of the following condition hold:*

- $V(S) \subseteq V(Q)$
- $E(S) \subseteq E(Q)$
- $\forall n \in V(S)$, $\mathcal{V}_S(n) \subseteq \mathcal{V}_Q(n)$
- $\forall e \in E(S)$, $\mathcal{E}_S(e) \subseteq \mathcal{E}_Q(e)$
- *If $|V(S)| = 1$, $n \in V(S)$ then there is no $d \in D$ s.t $\mathcal{V}_S(n) = \{d\}$*

b) Functional APIs

Definition 5 (roll-up) *Given a query Q and a node $n \in V(Q)$ **which is not a datatype node**, the operation roll-up(Q, n) encodes Q into a DL concept in $\mathcal{L}$ w.r.t n. The operation roll-up(Q, n) is defined as encode(Q, n, n) where encode is the recursive procedure described in Algorithm 1. We use roll-up(Q) as an abbreviation for roll-up(Q, o), where o is the root of Q*

Algorithm 1: Calculate encode(Q, n, m)

Input: a query Q and two nodes $n, m \in V(Q)$
Output: a concept C expressing Q in the DL language $\mathcal{L}$
$C \leftarrow c$, for some $c \in \mathcal{V}(n)$;
forall the $x \in \mathcal{V}(n)$ *such that* $x \neq c$ **do**
  $C \leftarrow C \sqcap x$
end
forall the $x \in V^1_{des,Q}(n)$ *such that* $x \neq m$ **do**
  if *x is not a datatype node* **then**
    $R \leftarrow \mathcal{E}(\langle n, x \rangle)$;
    $C \leftarrow C \sqcap R.encode(Q, x, n)$;
  end
  else
    $A \leftarrow \mathcal{E}(\langle n, x \rangle)$;
    $d \leftarrow \mathcal{V}(x)[0]$;
    $C \leftarrow C \sqcap \exists A.d$;
  end
end
if $n \neq o$ **then**
  Let $p \in V^1_{anc,Q}(n)$;
  if $p \neq m$ **then**
    $R \leftarrow \mathcal{E}(\langle p, n \rangle)$;
    $C \leftarrow C \sqcap \exists R^-.encode(Q, p, n)$;
  end
end
return C

c) Reasoning Services

Definition 6 (getAttributes) *The operation **getAttributes** takes as input a query Q and a focus node $n \in V(Q)$, n is not a datatype node and returns a set V, $V \subseteq A \times D$. Let $C = roll - up(Q, n)$ be the context of Q w.r.t n. Then a pair $\langle a, d \rangle$ belongs to V if and only if all of the following conditions are satisfied:*

- $K \not\models \exists a.d \sqsubseteq \bot$
- $K \not\models \exists a.d \sqsubseteq C$ *or* $K \models C \sqsubseteq \exists a.d$

d) Operations on Queries

Definition 7 (addAtribute) *Let Q be a query and let $n \in V(Q)$, n (not a datatype node) be a focus node. Then, for $\langle a, d\rangle \in getAttribute(Q, n)$, we define:*
$addAttribute(Q, n, \langle a, d\rangle) := Q \uplus R$ with R is a query such that :

- $V_R = \{n, n'\}$
- $E_R = \{\langle n, n'\rangle\}$
- $o = n$
- $\mathcal{V}_R(n) = \mathcal{V}_Q(n)$; $\mathcal{V}_R(n') = \{d\}$
- $\mathcal{E}_R(\langle n, n'\rangle) = a$

3 Querying over relation databases

Instead of using ABoxes, we would like to use a relational database DS to store data and then answer the query over this database. As a consequence, we need to map the relational database to a corresponding Description logic KB. We also have to make sure that the query refinement process can be done in description logic as in our framework while the query answering process can be performed in database level without any effect in the answers.

3.1 Mapping

Intuitively, the ideas of mapping are : a primary key will be map to a object identification and an table with n column will be map to n binary relation by mean of reification.
Consider a table T with n column $(n \geq 0)$: $C, C_1, ..., C_n$ where C is primary key column. T is corresponding to a concept A and n roles R_1, R_2, ..., R_n.

a) From DS to Description logic KB
We'll give the mapping rules and corresponding axioms in KB for following cases

i) There is no C_i such that C_i is foreign key column to a table.

$$
\begin{aligned}
&A(x) \equiv \exists y_1, .., y_n.T(x, y_1, ..., y_n)\\
&R(x, y) \equiv A(x) \wedge x = y\\
&R_1(x, y) \equiv A(x) \wedge \exists y_2, .., y_n.T(x, y, y_2, .., y_n)\\
&...\\
&R_i(x, y) \equiv A(x) \wedge \exists y_1..y_{i-1}y_{i+1}...y_n.T(x, y_1, ., y_{i-1}, y, y_{i+1}, .., y_n)\\
&...\\
&R_n(x, y) \equiv A(x) \wedge \exists y_1...y_{n-1}.T(x, y_1, .., y_{n-1}, y)
\end{aligned}
$$

Then, we have the following axioms in our KB

- $\exists R_i.\top \sqsubseteq A$ for $0 \leq i \leq n$
- $A \sqsubseteq \leq_1 R_i^{-1}.\top$ for $0 \leq i \leq n$

ii) There is a C_i such that C_i is foreign key column to a table named T_i
Call A_i is a corresponding concept of T_i

$$
\begin{aligned}
&A(x) \equiv \exists y_1, ..., y_n.T(x, y_1, ..., y_n) \\
&R(x, y) \equiv A(x) \wedge x = y \\
&R_1(x, y) \equiv A(x) \wedge \exists y_2, .., y_n.T(x, y, y_2, .., y_n) \\
&... \\
&R_i(x, y) \equiv A(x) \wedge \exists y_1..y_{i-1}y_{i+1}...y_n.T(x, y_1, ., y_{i-1}, y, y_{i+1}, .., y_n) \wedge A_i(y) \\
&... \\
&R_n(x, y) \equiv A(x) \wedge \exists y_1...y_{n-1}.T(x, y_1, .., y_{n-1}, y)
\end{aligned}
$$

Then, we have the following axioms in our KB

- $\exists R_j.\top \sqsubseteq A$ for $0 \leq j \leq n$
- $A \sqsubseteq \leq_1 R_j^{-1}.\top$ for $0 \leq j \leq n$
- $A \sqsubseteq \exists R_i.A_i$

b) From Description Logic KB to DS
Based on above section, we will prove that

i) There is no C_i such that C_i is foreign key column to a table.
$T(x, y_1, ..., y_n) \equiv A(x) \cap R_1(x, y_1) \cap ... \cap R_n(x, y_n)$

ii) There is a C_i such that C_i is foreign key column to a table named T_i
Call A_i is a corresponding concept of T_i
$T(x, y_1, ..., y_n) \equiv A(x) \cap R_1(x, y_1) \cap ... \cap R_n(x, y_n) \cap A_i(y_i)$

Proof

i) No foreign key
Based on the mapping rules, we have:

$$
\begin{aligned}
A(x) \cap R_1(x, y_1) \cap ... \cap R_n(x, y_n) \equiv &\exists y'_1, .., y'_n.T(x, y'_1, ..., y'_n) \\
&\cap \exists y'_2, .., y'_n.T(x, y_1, y'_2, .., y'_n) \\
&\cap ... \\
&\cap \exists y'_1...y'_{n-1}.T(x, y'_1, .., y'_{n-1}, y_n)
\end{aligned}
$$

- $\Rightarrow$ direction
It's trivial to see that:

$$
\begin{aligned}
T(x, y_1, .., y_n) \rightarrow &\exists y'_1, .., y'_n.T(x, y'_1, ..., y'_n) \\
&\cap \exists y'_2, .., y'_n.T(x, y_1, y'_2, .., y'_n) \\
&\cap ... \\
&\cap \exists y'_1...y'_{n-1}.T(x, y'_1, .., y'_{n-1}, y_n)
\end{aligned}
$$

Therefore: $T(x, y_1, .., y_n) \rightarrow A(x) \cap R_1(x, y_1) \cap ... \cap R_n(x, y_n)$

- $\Leftarrow$ direction
  Call I is a model of

$$\begin{aligned} P(x, y_1, y_2, .., y_n) = & \exists y'_1, .., y'_n . T(x, y'_1, ..., y'_n) \\ & \cap \exists y_{21}, .., y_{n1} . T(x, y_1, y_{21}, .., y_{n1}) \\ & \cap ... \\ & \cap \exists y_{1n} ... y_{(n-1)n} . T(x, y_{1n}, .., y_{(n-1)n}, y_n) \end{aligned}$$

  Assume that $T(x, y'_1, .., y'_n)^I$, $T(x, y_{1i}, .., y_i, y_{(i+1)i}, ..y_{ni})^I$ are true for all $1 \leq i \leq n$.
  Therefore $y'_i = y_i$ for all $1 \leq i \leq n$ under I because x is the primary key.
  Then we have $T(x, y_1, .., y_n)$ is true under I
  Then $P(x, y_1, .., y_n) \rightarrow T(x, y_1, .., y_n)$

ii) Has foreign key
  Can be proved analogously.

c) Preserving models
In this section, we will prove that DS and KB have the same FOL models.

Theorem 8 *If I is an FOL model of DS then I is an FOL model of corresponding KB*

Proof
Let T is an arbitrary table in DS. We have to prove that I is a model of corresponding axioms related to T
Consider 2 cases:

- No foreign key
  For all $(x^I, y_1^I, ..., y_n^I) \in T^I$, based on the mapping rules, we have
  - $x^I \in A^I$
  - $(x^I, y_i^I) \in R_i^I$ for all $1 \leq i \leq n$

  Moreover, according to the mapping rules $R_i(x, y) \rightarrow A(x)$, the axioms $\exists R_i . \top \sqsubseteq A$ is true under I too.

  Besides, assume that $(A \sqsubseteq \leq_1 R_i^{-1} . \top)$ is not true under I for an i. Therefore, with an x^I such that $x^I \in A^I$ there exists at least $y_a \neq y_b \in \Delta^I$ such that $(x^I, y_a) \in R_i^I$ and $(x^I, y_b) \in R_i^I$ (*). Since $x^I \in A^I$, there exists $y_1^I, y_2^I, ..., y_n^I$ such that $(x^I, y_1^I, ..., y_n^I) \in T^I$. Because x is a primary key, if $(x^I, y_1'^I, ..., y_n'^I) \in T^I$ then $y_i'^I = y_i^I$. Therefore from (*) and the corresponding mapping rule of R_i, we have $y_a = y_b \rightarrow$ contradiction. So the axiom is true under I

- Has Foreign key
  Can be proved analogously.

Theorem 9 *If I is an FOL model of KB then I is an FOL model of corresponding DS*

Proof

- No Foreign key
  First, assume that $x^I \in A^I$, $x^I, y_i^I \in R_i^I$ for all $1 \leq i \leq n$. We have to prove that $(x^I, y_1^I, .., y_n^I) \in T^I$
  According to the mapping rules :
  $T(x, y_1, .., y_n) = A(x) \cap R_1(x, y_1) \cap ... \cap R_n(x, y_n)$
  We have $(x^I, y_1^I, .., y_n^I) \in T^I$ trivially.

Second, assume that $x^I, y_1^I, .., y_n^I) \in T^I$ and $(x^I, y_1'^I, .., y_n'^I) \in T^I$ where $y_i^I \neq y_i'^I$ for some i. As a consequence, we have $(x^I, y_i^I) \in R_i$ and $(x^I, y_i'^I) \in R_i$. Since I is a model of KB then $y_i^I = y_i'^I \rightarrow$ Contradiction. Therefore the first argument of T is exactly the primary key of T

- Has Foreign key
  Can be proved analogously.

3.2 Queries and their refinement

a) Mapping of a query

Definition 10 *Consider a query Q according to the formal definition in Quelo framework. We will define the map function of Q inductively as follows :*

- *If Q is an atomic query, $A \in \mathcal{V}_Q(o)$; then $map(Q) = \exists y_1, .., y_n T(o, y_1, .., y_n)$ where T is corresponding table of A according to mapping rules*
- *If Q is a query which has only 1 node o and $\mathcal{V}_Q(o) = \{A_1, .., A_n\}$ then $map(Q) = map(Q_1) \wedge map(Q_2) \wedge ... \wedge map(Q_n)$ where Q_i is an atomic query with the root node corresponding to A_i*
- *If Q is a query which $V_Q = \{o, c_1, .., c_m\}$ and $\mathcal{E}(o, c_i) = R_i$ for $0 \leq i \leq n$ then*

$$\begin{aligned} map(Q) =& map(Q_1) \wedge ... \wedge map(Q_m) \wedge \\ & \exists y_{11}, .., y_{n_1 1} T_1(o, y_{11}, .., y_{n_1 1}) \wedge \\ & \exists y'_{11}, ..., y'_{n_1 1} T_1(o, y'_{11}, ..., y_{i_1 1}, ..., y'_{n_1 1}) \wedge \\ & ... \wedge \\ & \exists y_{1m}, .., y_{n_m m} T_m(o, y_{1m}, .., y_{n_m m}) \wedge \\ & \exists y'_{1m}, ..., y'_{n_m m} T_m(o, y'_{1m}, ..., y_{i_m m}, ..., y'_{n_m m}) \end{aligned}$$

where Q_i is the subquery of Q with the root c_i and T_i is the corresponding table of R_i according to the mapping rules

Theorem 11 *The answers of Q and $map(Q)$ are the same.*

Proof : by induction on the definition of $map(Q)$
Actually, $map(Q)$ is just a map of the corresponding conjunctive formula of Q according to the mapping rules. Therefore, they should have the same answer.

b) Query operations
First, we have: Q is corresponding to the set of subformulas $S(Q) = \{P_1, ..., P_n\}$ where $\bigcap P_i$ is the corresponding conjunctive query of Q and $map(Q)$ is corresponding to $S(map(Q)) = \{map(P_1), ..., map(P_n)\}$

i) Adding information (addRelation, addCompatibles)
Assume that we want to add another constraint to Q by a compatibles term or a relation, then we have $S(Q') = S(Q) \cup \{P_{n+1}\}$ where P_{n+1} is a concept expression $A(x)$ or role expression $R(x, y)$. $map(P_{n+1})$ is the map of P_{n+1} according to the mapping rules. Then $S(map(Q')) = S(map(Q)) \cup \{map(P_{n+1})\}$

ii) Removing information (delete, weakening)
Assume that we want to remove information P_i in $S(Q)$ then $S(Q') = S(Q) \backslash \{P_i\}$. $map(P_i)$ is the map of P_i according to the mapping rules. Then $S(map(Q')) = S(map(Q) \backslash \{map(P_i)\})$

4 Conclusion

To sum up, in this research we dealt with 2 goals :

- Extended Quelo framework with constants and attributes
- Defined the mapping rules between relational databases and description logic knowledge base which do not effect the query refinement process in Quelo.

In general, the first result help us consider more complicated queries in term of query building while the second result will support us answer these queries easily by using normal relational database techniques.

References

[Catarci, 2004] T. Catarci, T. D. Mascio, P. Dongilli, E. Franconi, G. Santucci, and S. Tes- saris, *AOBQ: An ontology based visual tool for query formulation support*, In Proceedings of the 16th Biennial European Conference on Artificial Inteligence (ECAI 2004), Valencia, Spain, 2004

[Paolo,2009] Guagliardo Paolo : *FSQ: Theoretical Foundations of an Ontology-Based Visual Tool for Query Formulation Support*, EMCL Master thesis, Free University of Bolzano, Italy, 2009

[Dongilli,2004] P. Dongilli, E. Franconi, and S. Tessaris : *SEWA: Semantics driven support for query formulation*, In Proceedings of the 2004 International Workshop on Description Logics, Whistler, BC, Canada, 2004