Containment of Conjunctive Object Meta-Queries

Andrea Calì, Michael Kifer

Faculty of Computer Science
Free University of Bolzano

State University of New York
at Stony Brook

XXXII Conference on Very Large Data Bases
VLDB 2006
Seoul, Korea, 15th September 2006
F-Logic (Frame-Logic)

F-Logic

- object-oriented formalism [Kifer & Lausen, JACM 1985]
- raised interest in the academia and commercially
  - building ontologies
  - reasoning in the Semantic Web
- meta-querying capability
- we will use a subset of F-Logic queries called F-Logic-Lite

Restrictions in F-Logic Lite

- no negation
- no default inheritance
- limited form of cardinality constraints
Query containment

- Well known problem in:
  1. query optimisation
  2. schema integration
  3. object classification (in DLs)
  4. service discovery
  5. ...

- amounts to check whether the result of a query is always contained in the result of another, for all databases

Query containment under constraints

- QC considering only databases that satisfy certain constraints

- relevant cases:
  1. functional and inclusion dependencies
  2. extended ER schemata
  3. Description Logic knowledge bases
Our contribution

1. we give a relational encoding of F-Logic Lite axioms in first-order rules
2. we consider containment of conjunctive meta-queries over relations encoding F-Logic Lite under the above rules
3. we provide a technique to decide query containment in such a case
4. we prove that checking containment is in NP
Outline

1. Introduction
2. Preliminaries
3. The encoding
4. Deciding containment by chasing
5. Complexity
6. Conclusions
F-Logic formalism by examples

Classes, subclasses and members

- \texttt{john:student} states that object \texttt{john} is a member of class \texttt{student};
- \texttt{freshman::student} and \texttt{student::person} state that class \texttt{freshman} is a subclass of the class \texttt{student} and \texttt{student} is a subclass of \texttt{person}

The above statements imply, for instance, that the following F-Logic formulae are true:

1. \texttt{john:person (john is a student)}
2. \texttt{freshman::person (class freshman is a subclass of person)}
Attributes

- **john[age->33]** means that object john has an attribute, age, whose value is 33;
- an attribute may have more than one value
Signature statements: type constraints

- `person[age=>number]` (type constraint) says that the attribute `age` of class `student` has the type `number`
- this type is inherited by subclasses and class instances of `person`
- this acts as a constraint on the statements of the form `john[age=>33]`
Signature statements: cardinality constraints

- `person[age \{0:1\} \*\Rightarrow \text{number}]` says that the attribute `age` has at most one value
- `person[name \{1:*\} \*\Rightarrow \text{string}]` says that the `name` attribute is mandatory in class `person`
A F-Logic feature

Classes are also objects

- statements like `student: class` are correct
- in this case `students` occurs as an object instead of a class
- it does not follow from this and the previous statements that `john: class`, `freshman: class`, or `student:: class`
Meta-queries examples

- Query ?- X::person. could have answers $X = \text{employee}$ and $X = \text{student}$

- Query ?- student[Att*=>string]. could have answers $\text{Attr} = \text{name}$ and $\text{Attr} = \text{major}$

- Query ?- student[Att*=>string], john[Att->Val]. asks for attributes of class student of type string that have a defined value for object john;
  - john does not need to be a member of student
Consider the meta-queries:

\[
q_1(A, B) :- T_1[A*=>T_2], T_2::T_3, T_3[B*=>\_].
\]

\[
q_2(A, B) :- T_1[A*=>T_2], T_2[B*=>\_].
\]

- \(q_1\) asks for pairs of attributes \(A, B\) s.t. the range of \(A\) is contained in the domain of \(B\)
- It is easy to see that \(q_1\) is contained in \(q_2\)
Low-level encoding of F-Logic Lite

- **member**(O, C): object O is a member of class C. This is the encoding for O : C.
- **sub**(C₁, C₂): class C₁ is a subclass of class C₂. This encodes the statement C₁ :: C₂.
- **data**(O, A, V): attribute A has value V on object O. This is the encoding for O[A->V].
- **type**(O, A, T): attribute A has type T for object O (recall that in F-logic classes are also objects). This encodes the statements of the form O[A*=>T].
- **mandatory**(A, O): attribute A is mandatory for object (class) O, i.e., it must have at least one value for O. This is an encoding of statements of the form O[A {1:*}*=>_].
- **funct**(A, O): A is a functional attribute for the object (class) O, i.e., it can have at most one value for O. This statement encodes O[A {0:1}*=>_].
Axioms

**Type correctness**

\[
\text{member}(V, T) :\neg \text{type}(O, A, T), \text{data}(O, A, V)
\]

**Subclass transitivity**

\[
\text{sub}(C_1, C_2) :\neg \text{sub}(C_1, C_3), \text{sub}(C_3, C_2)
\]

**Membership property**

\[
\text{member}(O, C_1) :\neg \text{member}(O, C), \text{sub}(C, C_1)
\]

**Functional attribute property**

\[
V = W :\neg \text{data}(O, A, V), \text{data}(O, A, W), \text{funct}(A, O).
\]

Notice that the equality predicate is used in the head.
### Axioms (contd.)

**Mandatory attributes definition**

\[
\forall O, A \exists V \text{ data}(O, A, V) \iff \text{mandatory}(A, O)
\]

Notice that this is **not** a Datalog rule: there is an existentially quantified variable in the head.

**Inheritance of types from classes to members**

\[
\text{type}(O, A, T) \iff \text{member}(O, C), \text{type}(C, A, T)
\]

**Inheritance of types from classes to subclasses**

\[
\text{sub}(C, C_1), \text{type}(C_1, A, T)
\]

**Supertyping**

\[
\text{type}(C, A, T) \iff \text{type}(C, A, T_1), \text{sub}(T_1, T)
\]
Axioms (contd.)

Inheritance of mandatory attributes to subclasses

\[
\text{mandatory}(A, C) \leftarrow \text{sub}(C, C_1), \text{mandatory}(A, C_1)
\]

Inheritance of mandatory attributes from classes to their members

\[
\text{mandatory}(A, O) \leftarrow \text{member}(O, C), \text{mandatory}(A, C)
\]

Inheritance of functional property to subclasses

\[
\text{funct}(A, C) \leftarrow \text{sub}(C, C_1), \text{funct}(A, C_1)
\]

Inheritance of functional property to members

\[
\text{funct}(A, O) \leftarrow \text{member}(O, C), \text{funct}(A, C)
\]
Meta-query containment

We denote the set of rules by $\Sigma_{FL}$

Meta-queries are conjunctive queries over the predicates encoding our formalism

Given two (meta)-queries $q_1$ and $q_2$, we say that $q_1$ is contained in $q_2$ with respect to $\Sigma_{FL}$, denoted $q_1 \subseteq_{\Sigma_{FL}} q_2$, if for every database $B$ that satisfies $\Sigma_{FL}$ we have $q_1(B) \subseteq q_2(B)$

$q(B)$ denotes the result of query $q$ on $B$
Chasing queries

Axioms that encode F-Logic Lite are tuple-generating dependencies (TGDs) and equality-generating dependencies (EGDs)

Chase for such classes of queries is known [Fagin et al. ICDT 2003]

Chasing wrt a TGD generates a new conjunct in the query

Chasing wrt an EGD equals two symbols (a variable and a constant or two variables)

the chase fails if chasing wrt an EGD equals two constants
Chasing and chase graph: example

Query to chase

\[ q() : \text{mandatory}(A, T), \text{type}(T, A, T) \]
This is derived from [Fagin et al. ICDT 2003]

**Theorem**

Given two conjunctive meta-queries $q_1$ and $q_2$, we have $q_1 \subseteq_{\Sigma_{FL}} q_2$ if and only if there exists a homomorphism that sends the conjuncts of $\text{body}(q_2)$ to conjuncts in $\text{chase}_{\Sigma_{FL}}(q_1)$ and $\text{head}(q_2)$ to $\text{head}(\text{chase}_{\Sigma_{FL}}(q_1))$

- $\text{chase}_{\Sigma_{FL}}(q_1)$ is the chase of $q_1$ wrt $\Sigma_{FL}$
- A homomorphism is a function that sends constants into themselves (and variables to variables or constants), preserving the structure of the predicates
- $\text{head}(\text{chase}_{\Sigma_{FL}}(q_1))$ is the head of $q_1$, possibly altered by chasing $q_1$
Due to an existentially quantified variable in the head of one of the rules, the chase might be infinite.

The previous property does not provide an algorithm for deciding containment.

**Plan of attack:**

1. Prove that if there is a homomorphism from $q_2$ to $\text{chase}_{\Sigma_{FL}}(q_1)$ with the desired properties, there is another from $q_2$ to a finite segment of $\text{chase}_{\Sigma_{FL}}(q_1)$.

2. Provide an upper bound (max no. of levels) for the above segment, depending on the queries.

3. Show that we can check containment by guessing a homomorphism from $q_2$ to the finite segment.
How to construct the chase

- First we chase wrt all rules except for the one that "invents" a fresh value (∃ in the head)
- We consider all the conjuncts obtained in this way as a new query (level 0)
- Then, we chase such query
- ... all this for technical reasons
Infinite chase

The only way to have an infinite chase is to have in $q_1$ a set of conjuncts of the form

\[
\text{mandatory}(A_1, T_1) \\
\text{type}(T_1, A_1, T_2) \\
\ldots \\
\text{mandatory}(A_{k-1}, T_{k-1}) \\
\text{type}(T_{k-1}, A_{k-1}, T_k) \\
\text{mandatory}(A_k, T_k) \\
\text{type}(T_k, A_k, T_1)
\]
Locality of the chase

- In the chase with TGDs, conjuncts are added according to more than one existing conjuncts.
- However, the chase enjoys **locality** properties:
  - Conjuncts at level 0 act as a map, driving the chase.
  - Every added conjunct is due to a conjunct at level 0 and another (with minor exceptions).
  - If we consider only the latter, we have **paths** in the graph as for IDs; such paths are called **primary**.
  - Due to the application of some rules, primary paths may branch.
Proving decidability

- Assume there is a homomorphism $\mu$ from $q_2$ to $\text{chase}_{\Sigma_{FL}}(q_1)$ with the desired properties.
- Consider a graph (forest) of the image of $q_2$ wrt $\mu$, considering the primary paths among them and the conjuncts where branching happens.
Regularity

- Primary paths evolve according to “regular” patterns
- Therefore, it is possible to excise the paths between adjacent nodes until they cover $2 \cdot |q_1|$ levels or less
- After every excision, the obtained conjuncts are still the image of $q_2$ wrt some homomorphism
Main result

Consider queries \( q_1, q_2 \); if there is a homomorphism from \( q_2 \) to \( chase_{\Sigma_{FL}}(q_1) \) with the desired properties, there is another from \( q_2 \) to a set of conjuncts in \( chase_{\Sigma_{FL}}(q_1) \) such that none of these conjuncts is at level greater than \( 2 \cdot |q_1| \).
Checking containment of F-Logic Lite meta-queries can be decided by a nondeterministic algorithm that runs in polynomial time in $|q_1|$ and $|q_2|$. 

Proof by guessing $|q_2|$ conjuncts in the first $2 \cdot |q_1|$ levels of $\text{chase}_{\Sigma_{FL}}(q_1)$
Conclusions

Wrap-up

- F-Logic is a popular tool for building ontologies
- We considered a relevant subset called F-Logic Lite
- Relational encoding
- Meta-query containment by chasing
- Complexity result

Future work

- Tight lower complexity bound
- More expressive query languages
- Finding a more general class of queries for which the same techniques apply