Formal Foundations: Conjunctive Queries and Datalog

Prof. Dr. Peter Fischer

Web Science
Universität Freiburg

SS 2012

Background

- Lecture so far:
  - "Rich" query languages for specific application domains
  - High level of complexity and expressiveness
  - Features before semantics and complexity
- Remainder of lecture
  - Toy languages with limited expressives
  - Strong semantics and complexity understanding
  - Desirable properties (e.g. containment decidable)
  - Results applicable to (subsets of) "rich" languages

Areas to study

- Conjunctive Queries: simple, core of most query languages, very well studied
- Query containment: classical query optimization problem
- Datalog: Recursion, gradual increase of expressiveness

Relational Model

- We assume that a countably infinite set **attr** of *attributes* is fixed.
- The *domain* is a countably infinite set **dom** (disjoint from **attr**).
- A *constant* is an element of **dom**.
- A relation schema is simply a relation name R, with $arity(R) = n$ (written as $R[n]$).
- A database schema is a nonempty finite set $\mathcal{R}$ of relation names.
- A *tuple* over a (possibly empty) finite set U of attributes (or over a relation schema $R[U]$) is a total mapping u from U to **dom**.
- or, a *tuple* is an ordered n-tuple ($n \geq 0$) of constants – an element of the Cartesian product $\mathbf{dom}^n$.
- A *database instance* is a finite set $\mathcal{I}$ that is the union of tuples.

Conjunctive Queries

Definition

A *conjunctive Query* Q over a database schema $\mathcal{R}$ is given as

$$ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}),$$

such that for $1 \leq i \leq n$

- R_i a relation name in $\mathcal{R}$ and
- $\vec{U}$ and $\vec{U_i}$ vectors of variables and constants;
- any variable appearing in $\vec{U}$ appears also in some $\vec{U_i}$.
- Left to $\leftarrow$ is the *head* of the query, and to the right there is the *body*. The atoms in the body are also called *subgoals*.

Example

$$\begin{gathered} Sales(Part,\ Supplier,\ Customer), \\ Part(PName,\ Type), \\ Cust(CName,\ CAddr), \\ Supp(SName,\ SAddr). \end{gathered}$$

$$Q: \qquad ans(T) \leftarrow Sales(P, S, C), Part(P, T), Cust(C, A), Supp(S, A)$$

$$ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}).$$

Answer

- The set of answers Q w.r.t. an instance $\mathcal{I}$ is denoted $Q(\mathcal{I})$.
- If there is a substitution (mapping) σ from the variables in $\vec{U_1}, \ldots, \vec{U_n}$ to the constants in **dom**, such that $\sigma(R_1(\vec{U_1})), \ldots, \sigma(R_n(\vec{U_n})) \in \mathcal{I}$, then by applying the same substitution σ to $\vec{U}$, we say that $\sigma(ans(\vec{U}))$ is an answer in $Q(\mathcal{I})$.
- Note that a substitution is a function such that a variable is mapped into only one constant, and a constant is mapped into itself.

$$ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}).$$

Complexity of Query Answering

Let Q be a conjunctive query and $\mathcal{I}$ a database instance. what is the complexity of computing all the answers of $Q(\mathcal{I})$?

$$N^m$$

where N is the size of $\mathcal{I}$ (number of constants in $\mathcal{I}$), and m the size of the query (number of distinct variables in Q).

Boolean Conjunctive Query

$$true \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}).$$

- Boolean conjunctive query answering is a decision problem.
- The complexity of the boolean conjunctive query answering is NP-complete.

Decision Problems

- Problems where the answer is “yes” or “no”
- Formally,
  - A language L over some alphabet Σ.
  - An *instance* is given as a word $x \in \Sigma^*$.
  - Question: whether $x \in L$ holds
- The resources (i.e., either time or space) required in the worst case to find the correct answer for any instance x of a problem L is referred to as the *complexity* of the problem L

NP-completeness proof

- The problem is in NP:
  - Guess a substitution (mapping) from all the variables in Q to a set of constants in $\mathcal{I}$,
  - Check whether the substitution makes the subgoals in the body true.
- The problem is NP hard: reduction from 3-SAT.

$$(v_1 \vee v_2 \vee \bar{v}_3) \wedge (v_1 \vee \bar{v}_2 \vee \bar{v}_4) \wedge (\bar{v}_1 \vee v_3 \vee v_4)$$

NP-completeness proof

The problem is NP hard: reduction from 3-SAT.

$$(v_1 \vee v_2 \vee \bar{v_3}) \wedge (v_1 \vee \bar{v_2} \vee \bar{v_4}) \wedge (\bar{v_1} \vee v_3 \vee v_4)$$

$\mathcal{I}$: $r(1,1,1)$, $r(1,1,0)$, $r(1,0,1)$, $r(1,0,0)$, $r(0,1,1)$, $r(0,1,0)$, $r(0,0,1)$, $c(1,0)$, $c(0,1)$.
Q: $r(v_1, v_2, nv_3)$, $r(v_1, nv_2, nv_4)$, $r(nv_1, v_3, v_4)$, $c(v_1, nv_1)$, $c(v_2, nv_2)$, $c(v_3, nv_3)$, $c(v_4, nv_4)$.

Interesting Problems

Let Q, Q_1, Q_2 be conjunctive queries.

Containment: $Q_1 \sqsubseteq Q_2$, i.e., $Q_1(\mathcal{I}) \subseteq Q_2(\mathcal{I})$ for any instance $\mathcal{I}$?

Equivalence: $Q_1 \equiv Q_2$, i.e., $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$?

Minimization: Given Q_1, construct an equivalent query Q_2, which has as most as much subgoals in its body as Q_1 and is minimal in the sense, that any query Q_3 being equivalent to Q_2 has at least as much subgoals in the body as Q_2.

Q_2 is called *minimal*.

Example

$$
\begin{gathered}
Sales(Part,\ Supplier,\ Customer),\\
Part(PName,\ Type),\\
Cust(CName,\ CAddr),\\
Supp(SName,\ SAddr).
\end{gathered}
$$

Equivalent queries:

$$
\begin{aligned}
Q: \quad & ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A)\\
Q': \quad & ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A),\\
& \qquad Sales(P',S',C'), Part(P',T)
\end{aligned}
$$

Motivation and Applications

- Fundamentally interesting problem
- Enables removing redundant subgoals
- Validate query transformations (i.e. possible optimizations)
- Detecting independence of queries from updates
- Semantic caching
- Answering queries using views

Lemma

Let

$$\begin{array}{ll} Q_1: & ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}) \\ Q_2: & ans(\vec{U}) \leftarrow S_1(\vec{V_1}), \ldots, S_m(\vec{V_m}) \end{array}$$

be conjunctive queries, where

$$\{R_1(\vec{U_1}), \ldots, R_n(\vec{U_n})\} \supseteq \{S_1(\vec{V_1}), \ldots, S_m(\vec{V_m})\}$$

Then $Q_1 \sqsubseteq Q_2$.

Substitution

- A *substitution* θ over a set of variables $\mathcal{V}$ is a mapping from $\mathcal{V}$ to $\mathcal{V} \cup \mathbf{dom}$, where **dom** a corresponding domain.
- We extend θ to constants $a \in \mathbf{dom}$ and relation names $R \in \mathcal{R}$, where $\theta(a) = a$, resp. $\theta(R) = R$.

Example

Consider

$$Q: \quad ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A)$$

$$Q': \quad ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A), Sales(P',S',C'), Part(P',T)$$

and θ:

X	P	P'	S	S'	C	C'	T	A
$\theta(X)$	P	P	S	S	C	C	T	A

Containment Mapping

Given conjunctive queries

$$\begin{aligned} Q_1 : \quad & ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}) \\ Q_2 : \quad & ans(\vec{V}) \leftarrow S_1(\vec{V_1}), \ldots, S_m(\vec{V_m}) \end{aligned}$$

Substitution θ is called *containment mapping* from Q_2 to Q_1, if Q_2 can be transformed by means of θ to become Q_1:

- $\theta(ans(\vec{V})) = ans(\vec{U})$,
- for $i = 1, \ldots, m$ there exists a $j \in \{1, \ldots, n\}$, such that $\theta(S_i(\vec{V_i})) = R_j(\vec{U_j})$.

Example

$$Q : \quad ans(T) \leftarrow Sales(P, S, C), Part(P, T), Cust(C, A), Supp(S, A)$$

$$Q' : \quad ans(T) \leftarrow Sales(P, S, C), Part(P, T), Cust(C, A), Supp(S, A), Sales(P', S', C'), Part(P', T)$$

θ:

X	P	P'	S	S'	C	C'	T	A
$\theta(X)$	P	P	S	S	C	C	T	A

θ is a containment mapping.

Theorem

Let

$$\begin{aligned} Q_1 : \quad & ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n}) \\ Q_2 : \quad & ans(\vec{V}) \leftarrow S_1(\vec{V_1}), \ldots, S_m(\vec{V_m}) \end{aligned}$$

be conjunctive queries.
$Q_1 \sqsubseteq Q_2$ iff there exists a containment mapping θ from Q_2 to Q_1.

Proof "$\Leftarrow$":

Intuition:

- Given the containment mapping, and a substitution that proves t $\in$ Q1, we can construct a substitution to prove t $\in$ Q2
- Q2 may have more answers than Q1 because Q1 may have additional subgoals that further restrict its answers

Formal:
There exists containment mapping θ.
Let $\mathcal{I}$ be an instance of Q_1 and let $\mu \in Q_1(\mathcal{I})$.
There exists a substitution τ, such that $\tau(\vec{U}_j) \in \mathcal{I}(R_j)$, $j \in \{1, \ldots, n\}$ and $\mu = \tau(\vec{U})$.
Consider a substitution $\tau' = \tau \circ \theta$ and further $\tau'(S_i(\vec{V}_i))$.
There holds $\tau'(\vec{V}_i) \in \mathcal{I}(S_i)$, $i \in \{1, \ldots, m\}$ and therefore also $\mu = \tau'(\vec{V})$.
D.h., $\mu \in Q_2(\mathcal{I})$.

Canonical Instance

Let Q be a conjunctive $ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n})$ over a database schema $\mathcal{R}$. The *canonical instance* $\mathcal{I}_Q$ to Q is constructed as follows.

$\mathcal{I}_Q$ is an instance of $\mathcal{R} = \{R_1, \ldots, R_n\}$.

Let τ be a substitution, which assigns to any X in Q an unique constant a_X.

- For any literal $R(t_1, \ldots, t_n)$ in the body, insert a tupel of the form $(\tau(t_1), \ldots, \tau(t_n))$ into $\mathcal{I}_Q(R)$; we also write $\tau(R(t_1, \ldots, t_n)) \in \mathcal{I}_Q(R)$.

No other tuples are inserted into $\mathcal{I}_Q(R)$.

τ is called *canonical substitution*.
Intuition: Frozen conjunctive query

- Create unique constant for each variable in Q
- Database only contains the subgoals of Q in a "frozen" form

Example

$$Q: \quad ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A)$$

$$Q': \quad ans(T) \leftarrow Sales(P,S,C), Part(P,T), Cust(C,A), Supp(S,A), Sales(P',S',C'), Part(P',T)$$

$\mathcal{I}_Q$:

Sales

a_P	a_S	a_C

Part

a_P	a_T

Cust

a_C	a_A

Supp

a_S	a_A

$\mathcal{I}_{Q'}$:

Sales

a_P	a_S	a_C
$a_{P'}$	$a_{S'}$	$a_{C'}$

Part

a_P	a_T
$a_{P'}$	a_T

Cust

a_C	a_A

Supp

a_S	a_A

Proof "$\Rightarrow$":

$Q_1 \sqsubseteq Q_2$.
Consider $\mathcal{I}_{Q_1}$ and the corresponding canonical substitution τ.
Then $\tau(ans(\vec{U})) \in Q_1(\mathcal{I}_{Q_1})$.
Because of $Q_1 \sqsubseteq Q_2$ further $\tau(ans(\vec{U})) \in Q_2(\mathcal{I}_{Q_1})$.
Thus, there exists a substitution ρ, such that $\rho(S_i(\vec{V}_i)) = \tau(R_j(\vec{U}_j))$, $1 \leq i \leq m$, $j \in \{1, \ldots, n\}$ und $\rho(ans(\vec{V})) = \tau(ans(\vec{U}))$.
$\tau^{-1} \circ \rho$ is a containment mapping.

Corollary

Let

$$Q_1: \quad ans(\vec{U}) \leftarrow R_1(\vec{U_1}), \ldots, R_n(\vec{U_n})$$
$$Q_2: \quad ans(\vec{V}) \leftarrow S_1(\vec{V_1}), \ldots, S_m(\vec{V_m})$$

be conjunctive queries, $\mathcal{I}_{Q_1}$ the canonical instance to Q_1 with canonical substitution τ.
$Q_1 \sqsubseteq Q_2$, iff $\tau(ans(\vec{U})) \in Q_2(\mathcal{I}_{Q_1})$.

Proof: We show, whenever $\tau(ans(\vec{U})) \in Q_2(\mathcal{I}_{Q_1})$, then $Q_1 \sqsubseteq Q_2$.
For any S_j in Q_2, $\mathcal{I}_{Q_1}$ is not empty. Therefore, for S_j there exists a R_i, such that $S_j = R_i$. Further, there exists a substitution ρ, such that for $S_j(\vec{V_j})$ we have $\rho(V_j) \in \mathcal{I}_{Q_1}(R_i)$. $\rho \circ \tau^{-1}$ is a containment mapping from Q_2 to Q_1.

Example

$$ans(a_T) \in Q(\mathcal{I}_{Q'})$$

and

$$ans(a_T) \in Q'(\mathcal{I}_Q).$$