Conjunctive Queries

Formal Methods

Giuseppe De Giacomo

Sapienza Università di Roma
MSc in Engineering in Computer Science

Conjunctive queries (CQs)

Def.: A conjunctive query (CQ) is a FOL query of the form

$$\exists \vec{y}.conj(\vec{x}, \vec{y})$$

where $conj(\vec{x}, \vec{y})$ is a conjunction (i.e., an "and") of atoms and equalities, over the free variables $\vec{x}$, the existentially quantified variables $\vec{y}$, and possibly constants.

Note:

- CQs contain no disjunction, no negation, no universal quantification, and no function symbols besides constants.
- Hence, they correspond to relational algebra select-project-join (SPJ) queries.
- CQs are the most frequently asked queries.

Conjunctive queries and SQL – Example

Relational alphabet:
Person(name, age), Lives(person, city), Manages(boss, employee)

Query: find the name and the age of the persons who live in the same city as their boss.

Conjunctive queries and SQL – Example

Relational alphabet:
Person(name, age), Lives(person, city), Manages(boss, employee)

Query: find the name and the age of the persons who live in the same city as their boss.

Expressed in SQL:

```
SELECT P.name, P.age
FROM Person P,  Manages M,  Lives L1,  Lives L2
WHERE P.name = L1.person  AND  P.name = M.employee  AND
      M.boss = L2.person  AND  L1.city = L2.city
```

Conjunctive queries and SQL – Example

Relational alphabet:
Person(name, age), Lives(person, city), Manages(boss, employee)

Query: find the name and the age of the persons who live in the same city as their boss.

Expressed in SQL:

```
SELECT P.name, P.age
FROM Person P,  Manages M,  Lives L1,  Lives L2
WHERE P.name = L1.person  AND  P.name = M.employee  AND
      M.boss = L2.person  AND  L1.city = L2.city
```

Expressed as a CQ:

$$\exists b, e, p_1, c_1, p_2, c_2.\text{Person}(n, a) \wedge \text{Manages}(b, e) \wedge \text{Lives}(p1, c1) \wedge \text{Lives}(p2, c2) \wedge n = p1 \ \wedge \ n = e \ \wedge \ b = p2 \ \wedge \ c1 = c2$$

Conjunctive queries and SQL – Example

Relational alphabet:
Person(name, age), Lives(person, city), Manages(boss, employee)

Query: find the name and the age of the persons who live in the same city as their boss.

Expressed in SQL:

```
SELECT P.name, P.age
FROM Person P,  Manages M,  Lives L1,  Lives L2
WHERE P.name = L1.person  AND  P.name = M.employee  AND
      M.boss = L2.person  AND  L1.city = L2.city
```

Expressed as a CQ:

$$\exists b, e, p_1, c_1, p_2, c_2.\text{Person}(n, a) \wedge \text{Manages}(b, e) \wedge \text{Lives}(p1, c1) \wedge \text{Lives}(p2, c2) \wedge n = p1 \ \wedge \ n = e \ \wedge \ b = p2 \ \wedge \ c1 = c2$$

Or simpler: $\exists b, c.\text{Person}(n, a) \wedge \text{Manages}(b, n) \wedge \text{Lives}(n, c) \wedge \text{Lives}(b, c)$

Datalog notation for CQs

A CQ $q = \exists \vec{y}.conj(\vec{x}, \vec{y})$ can also be written using datalog notation as

$$q(\vec{x}_1) \leftarrow conj'(\vec{x}_1, \vec{y}_1)$$

where $conj'(\vec{x}_1, \vec{y}_1)$ is the list of atoms in $conj(\vec{x}, \vec{y})$ obtained by equating the variables $\vec{x}$, $\vec{y}$ according to the equalities in $conj(\vec{x}, \vec{y})$.

As a result of such an equality elimination, we have that $\vec{x}_1$ and $\vec{y}_1$ can contain constants and multiple occurrences of the same variable.

Def.: In the above query q, we call:

- $q(\vec{x}_1)$ the head;
- $conj'(\vec{x}_1, \vec{y}_1)$ the body;
- the variables in $\vec{x}_1$ the distinguished variables;
- the variables in $\vec{y}_1$ the non-distinguished variables.

Conjunctive queries – Example

- Consider an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, E^{\mathcal{I}})$, where $E^{\mathcal{I}}$ is a binary relation – *note that such interpretation is a (directed) graph*.
- The following CQ q returns all nodes that participate to a triangle in the graph:

$$\exists y, z. E(x, y) \wedge E(y, z) \wedge E(z, x)$$

- The query q in datalog notation becomes:

$$q(x) \leftarrow E(x, y), E(y, z), E(z, x)$$

- The query q in SQL is (we use `Edge(f,s)` for $E(x, y)$):

```
SELECT E1.f
FROM Edge E1, Edge E2, Edge E3
WHERE E1.s = E2.f AND E2.s = E3.f AND E3.s = E1.f
```

Nondeterministic evaluation of CQs

Since a CQ contains only existential quantifications, we can evaluate it by:

1. guessing a truth assignment for the non-distinguished variables;
2. evaluating the resulting formula (that has no quantifications).

```
boolean ConjTruth(𝓘,α,∃y⃗.conj(x⃗,y⃗)) {
  GUESS assignment α[y⃗ ↦ a⃗] {
     return Truth(𝓘,α[y⃗ ↦ a⃗],conj(x⃗,y⃗));
}
```

where Truth$(\mathcal{I}, \alpha, \varphi)$ is defined as for FOL queries, considering only the required cases.

Nondeterministic CQ evaluation algorithm

```
boolean Truth(𝓘,α,φ) {
  if (φ is t_1 = t_2)
    return TermEval(𝓘,α,t_1) = TermEval(𝓘,α,t_2);
  if (φ is P(t_1,...,t_k))
    return P^𝓘(TermEval(𝓘,α,t_1),...,TermEval(𝓘,α,t_k));
  if (φ is ψ ∧ ψ')
    return Truth(𝓘,α,ψ) ∧ Truth(𝓘,α,ψ');
}

Δ^𝓘 TermEval(𝓘,α,t) {
   if (t is a variable x) return α(x);
   if (t is a constant c) return c^𝓘;
}
```

CQ evaluation – Combined, data, and query complexity

Theorem (Combined complexity of CQ evaluation)

$\{\langle \mathcal{I}, \alpha, q\rangle \mid \mathcal{I}, \alpha \models q\}$ *is NP-complete — see below for hardness*

- *time: exponential*
- *space: polynomial*

Theorem (Data complexity of CQ evaluation)

$\{\langle \mathcal{I}, \alpha\rangle \mid \mathcal{I}, \alpha \models q\}$ *is* LogSpace

- *time: polynomial*
- *space: logarithmic*

Theorem (Query complexity of CQ evaluation)

$\{\langle \alpha, q\rangle \mid \mathcal{I}, \alpha \models q\}$ *is NP-complete — see below for hardness*

- *time: exponential*
- *space: polynomial*

3-colorability

A graph is k-colorable if it is possible to assign to each node one of k colors in such a way that every two nodes connected by an edge have different colors.

Def.: 3-colorability is the following decision problem

Given a graph $G = (V, E)$, is it 3-colorable?

Theorem

3-colorability is NP-complete.

3-colorability

A graph is *k-colorable* if it is possible to assign to each node one of k colors in such a way that every two nodes connected by an edge have different colors.

Def.: 3-colorability is the following decision problem

Given a graph $G = (V, E)$, is it 3-colorable?

Theorem

3-colorability is NP-complete.

We exploit 3-colorability to show NP-hardness of conjunctive query evaluation.

Reduction from 3-colorability to CQ evaluation

Let $G = (V, E)$ be a graph. We define:

- An Interpretation: $\mathcal{I} = (\Delta^{\mathcal{I}}, E^{\mathcal{I}})$ where:
  - $\Delta^{\mathcal{I}} = \{\mathrm{r}, \mathrm{g}, \mathrm{b}\}$
  - $E^{\mathcal{I}} = \{(\mathrm{r}, \mathrm{g}), (\mathrm{g}, \mathrm{r}), (\mathrm{r}, \mathrm{b}), (\mathrm{b}, \mathrm{r}), (\mathrm{g}, \mathrm{b}), (\mathrm{b}, \mathrm{g})\}$
- A conjunctive query: Let $V = \{x_1, \ldots, x_n\}$, then consider the boolean conjunctive query defined as:

$$q_G \;=\; \exists x_1, \ldots, x_n. \bigwedge_{(x_i, x_j) \in E} E(x_i, x_j) \wedge E(x_j, x_i)$$

Theorem

G is 3-colorable iff $\mathcal{I} \models q_G$.

NP-hardness of CQ evaluation

The previous reduction immediately gives us the hardness for combined complexity.

Theorem
CQ evaluation is NP-hard in combined complexity.

NP-hardness of CQ evaluation

The previous reduction immediately gives us the hardness for combined complexity.

Theorem
CQ evaluation is NP-hard in combined complexity.

Note: in the previous reduction, the interpretation does not depend on the actual graph. Hence, the reduction provides also the lower-bound for query complexity.

Theorem
CQ evaluation is NP-hard in query (and combined) complexity.

Recognition problem and boolean query evaluation

Consider the recognition problem associated to the evaluation of a query q of arity k. Then

$$\mathcal{I}, \alpha \models q(x_1, \ldots, x_k) \qquad \text{iff} \qquad \mathcal{I}_{\alpha,\vec{c}} \models q(c_1, \ldots, c_k)$$

where $\mathcal{I}_{\alpha,\vec{c}}$ is identical to $\mathcal{I}$ but includes new constants $c_1, \ldots, c_k$ that are interpreted as $c_i^{\mathcal{I}_{\alpha,\vec{c}}} = \alpha(x_i)$.

That is, we can reduce the recognition problem to the evaluation of a boolean query.

Homomorphism

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, c^{\mathcal{I}}, \ldots)$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, P^{\mathcal{J}}, \ldots, c^{\mathcal{J}}, \ldots)$ be two interpretations over the same alphabet (for simplicity, we consider only constants as functions).

Def.: A homomorphism from $\mathcal{I}$ to $\mathcal{J}$

is a mapping $h : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that:

- $h(c^{\mathcal{I}}) \ = \ c^{\mathcal{J}}$
- $(o_1, \ldots, o_k) \in P^{\mathcal{I}}$ implies $(h(o_1), \ldots, h(o_k)) \in P^{\mathcal{J}}$

Note: An isomorphism is a homomorphism that is one-to-one and onto.

Theorem

FOL is unable to distinguish between interpretations that are isomorphic.

Proof. See any standard book on logic. □

Canonical interpretation of a (boolean) CQ

Let q be a conjunctive query $\exists x_1, \ldots, x_n.conj$

Def.: The canonical interpretation $\mathcal{I}_q$ associated with q

is the interpretation $\mathcal{I}_q = (\Delta^{\mathcal{I}_q}, P^{\mathcal{I}_q}, \ldots, c^{\mathcal{I}_q}, \ldots)$, where

- $\Delta^{\mathcal{I}_q} = \{x_1, \ldots, x_n\} \cup \{c \mid c \text{ constant occurring in } q\}$, i.e., all the variables and constants in q;
- $c^{\mathcal{I}_q} = c$, for each constant c in q;
- $(t_1, \ldots, t_k) \in P^{\mathcal{I}_q}$ iff the atom $P(t_1, \ldots, t_k)$ occurs in q.

Canonical interpretation of a (boolean) CQ – Example

Consider the boolean query q

$$q(c) \leftarrow E(c,y), E(y,z), E(z,c)$$

Then, the canonical interpretation $\mathcal{I}_q$ is defined as

$$\mathcal{I}_q = (\Delta^{\mathcal{I}_q}, E^{\mathcal{I}_q}, c^{\mathcal{I}_q})$$

where

- $\Delta^{\mathcal{I}_q} = \{y, z, c\}$
- $E^{\mathcal{I}_q} = \{(c,y), (y,z), (z,c)\}$
- $c^{\mathcal{I}_q} = c$

Homomorphism theorem

Theorem ([CM77])

For boolean CQs, $\mathcal{I} \models q$ iff there exists a homomorphism from $\mathcal{I}_q$ to $\mathcal{I}$.

Proof.
"$\Rightarrow$" Let $\mathcal{I} \models q$, let α be an assignment to the existential variables that makes q true in $\mathcal{I}$, and let $\hat{\alpha}$ be its extension to constants. Then $\hat{\alpha}$ is a homomorphism from $\mathcal{I}_q$ to $\mathcal{I}$.

"$\Leftarrow$" Let h be a homomorphism from $\mathcal{I}_q$ to $\mathcal{I}$. Then restricting h to the variables only we obtain an assignment to the existential variables that makes q true in $\mathcal{I}$. $\square$

Illustration of homomorphism theorem – Interpretation

Consider the following interpretation $\mathcal{I}$:

- $\Delta^{\mathcal{I}} = \{john, paul, george, mick, ny, london, 0, \ldots, 110\}$
- $Person^{\mathcal{I}} = \{(john, 30), (paul, 60), (george, 35), (mick, 35)\}$
- $Lives^{\mathcal{I}} = \{(john, ny), (paul, ny), (george, london), (mick, london)\}$
- $Manages^{\mathcal{I}} = \{(paul, john), (george, mick), (paul, mick)\}$

In relational notation:

$Person^{\mathcal{I}}$

name	age
john	30
paul	60
george	35
mick	35

$Lives^{\mathcal{I}}$

name	city
john	ny
paul	ny
george	london
mick	london

$Manages^{\mathcal{I}}$

boss	emp. name
paul	john
george	mick
paul	mick

Illustration of homomorphism theorem – Query

Consider the following query q:

$$q() \leftarrow Person(john, z), Manages(x, john), Lives(x, y), Lives(john, y)$$

"There exists a manager that has john as an employee and lives in the same city of him?"

The canonical model $\mathcal{I}_q$ is:

- $\Delta^{\mathcal{I}} = \{john, x, y, z\}$
- $john^{\mathcal{I}} = john$
- $Person^{\mathcal{I}_q} = \{(john, z)\}$
- $Lives^{\mathcal{I}_q} = \{(john, y), (x, y)\}$
- $Manages^{\mathcal{I}_q} = \{(x, john)\}$

In relational notation:

$Person^{\mathcal{I}_q}$

name	age
john	z

$Lives^{\mathcal{I}_q}$

name	city
john	y
x	y

$Manages^{\mathcal{I}_q}$

boss	emp. name
x	john

Illustration of homomorphism theorem – If-direction

Hp: $\mathcal{I} \models q$. **Th**: There exists an homomrphism $h : \mathcal{I}_q \rightarrow \mathcal{I}$.
If $\mathcal{I} \models q$, then there exists an assignment $\hat{\alpha}$ such that $\langle \mathcal{I}, \alpha \rangle \models body(q)$:

- $\alpha(x) = paul$
- $\alpha(z) = 30$
- $\alpha(y) = ny$

Let us extend $\hat{\alpha}$ to constants:

- $\hat{\alpha}(john) = john$

$h = \hat{\alpha}$ is an homomorphism from $\mathcal{I}_{q_1}$ to $\mathcal{I}$:

- $h(john^{\mathcal{I}_q}) = john^{\mathcal{I}}$? Yes!
- $(john, z)) \in Person^{\mathcal{I}_q}$ implies $(h(john), h(z)) \in Person^{\mathcal{I}}$?
  Yes: $(john, 30) \in Person^{\mathcal{I}}$;
- $(john, x) \in Lives^{\mathcal{I}_q}$ implies $h(john), h(x)) \in Lives^{\mathcal{I}}$?
  Yes: $(john, ny) \in Lives^{\mathcal{I}}$;
- $(x, y) \in Lives^{\mathcal{I}_q}$ implies $(h(x), h(y)) \in Lives^{\mathcal{I}}$?
  Yes: $(paul, ny) \in Lives^{\mathcal{I}}$;
- $(x, john) \in Manages^{\mathcal{I}_q}$ implies $(h(x), h(john)) \in Manages^{\mathcal{I}}$?
  Yes: $(paul, john) \in Manages^{\mathcal{I}}$.

Illustration of homomorphism theorem – Only-if-direction

Hp: There exists an homomrphism $h : \mathcal{I}_q \to \mathcal{I}$. **Th**: $\mathcal{I} \models q$.
Let $h : \mathcal{I}_q \to \mathcal{I}$:

- $h(john) = john$;
- $h(x) = paul$;
- $h(z) = 30$;
- $h(y) = ny$.

Let us define an assignment α by restricting h to variables:

- $\alpha(x) = paul$;
- $\alpha(z) = 30$;
- $\alpha(y) = ny$.

Then $\langle \mathcal{I}, \alpha \rangle \models body(q)$. Indeed:

- $(john, \alpha(z)) = (john, 30) \in Person^{\mathcal{I}}$;
- $(\alpha(x), john) = (paul, john) \in Manages^{\mathcal{I}}$;
- $(\alpha(x), \alpha(y)) = (paul, ny) \in Lives^{\mathcal{I}}$;
- $(john, \alpha(y)) = (john, ny) \in Lives^{\mathcal{I}}$.

Canonical interpretation and (boolean) CQ evaluation

The previous result can be rephrased as follows:

(The recognition problem associated to) query evaluation can be reduced to finding a homomorphism.

Finding a homomorphism between two interpretations (aka relational structures) is also known as solving a Constraint Satisfaction Problem (CSP), a problem well-studied in AI – see also [KV98].

Observations

Theorem

$\mathcal{I}_q \models q$ is always true.

Proof. By Chandra Merlin theorem: $\mathcal{I}_q \models q$ iff there exists homomorph. from $\mathcal{I}_q$ to $\mathcal{I}_q$. Identity is one such homomorphism. $\square$

Theorem

Let h be a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$, and h' be a homomorphism from $\mathcal{I}_2$ to $\mathcal{I}_3$. Then $h \circ h'$ is a homomorphism form $\mathcal{I}_1$ to $\mathcal{I}_3$.

Proof. Just check that $h \circ h'$ satisfied the definition of homomorphism: i.e. $h'(h(\cdot))$is a mapping from $\Delta^{\mathcal{I}_1}$ to $\Delta^{\mathcal{I}_3}$ such that:

- $h'(h(c^{\mathcal{I}_1})) = c^{\mathcal{I}_3}$;
- $(o_1, \ldots, o_k) \in P^{\mathcal{I}_1}$ implies $(h'(h(o_1)), \ldots, h'(h(o_k))) \in P^{\mathcal{I}_3}$. $\square$

The CQs characterizing property

Def.: Homomorphic equivalent interpretations

Two interpretations $\mathcal{I}$ and $\mathcal{J}$ are homomorphically equivalent if there is homomorphism $h_{\mathcal{I},\mathcal{J}}$ from $\mathcal{I}$ to $\mathcal{J}$ and homomorphism $h_{\mathcal{J},\mathcal{I}}$ from $\mathcal{J}$ to $\mathcal{I}$.

Theorem (model theoretic characterization of CQs)

CQs are unable to distinguish between interpretations that are homomorphic equivalent.

Proof. Consider any two homomorphically equivalent interpretations $\mathcal{I}$ and $\mathcal{J}$ with homomorphism $h_{\mathcal{I},\mathcal{J}}$ from $\mathcal{I}$ to $\mathcal{J}$ and homomorphism $h_{\mathcal{J},\mathcal{I}}$ from $\mathcal{J}$ to $\mathcal{I}$.

- If $\mathcal{I} \models q$ then there exists a homomorphism h from $\mathcal{I}_q$ to $\mathcal{I}$. But then $h \circ h_{\mathcal{I},\mathcal{J}}$ is a homomorphism from $\mathcal{I}_q$ to $\mathcal{J}$, hence $\mathcal{J} \models q$.
- Similarly, if $\mathcal{J} \models q$ then there exists a homomorphism g from $\mathcal{I}_q$ to $\mathcal{J}$. But then $g \circ h_{\mathcal{J},\mathcal{I}}$ is a homomorphism from $\mathcal{I}_q$ to $\mathcal{I}$, hence $\mathcal{I} \models q$. $\square$

Query containment

Def.: Query containment

Given two FOL queries φ and ψ of the same arity, φ is contained in ψ, denoted $\varphi \subseteq \psi$, if for all interpretations $\mathcal{I}$ and all assignments α we have that

$$\mathcal{I}, \alpha \models \varphi \quad \text{implies} \quad \mathcal{I}, \alpha \models \psi$$

(In logical terms: $\varphi \models \psi$.)

Note: Query containment is of special interest in query optimization.

Query containment

Def.: Query containment

Given two FOL queries φ and ψ of the same arity, φ is contained in ψ, denoted $\varphi \subseteq \psi$, if for all interpretations $\mathcal{I}$ and all assignments α we have that

$$\mathcal{I}, \alpha \models \varphi \quad \text{implies} \quad \mathcal{I}, \alpha \models \psi$$

(In logical terms: $\varphi \models \psi$.)

Note: Query containment is of special interest in query optimization.

Theorem

For FOL queries, query containment is undecidable.

Proof.: Reduction from FOL logical implication. $\square$

Query containment for CQs

For CQs, query containment $q_1(\vec{x}) \subseteq q_2(\vec{x})$ can be reduced to query evaluation.

1. Freeze the free variables, i.e., consider them as constants.
   This is possible, since $q_1(\vec{x}) \subseteq q_2(\vec{x})$ iff
   - $\mathcal{I}, \alpha \models q_1(\vec{x})$ implies $\mathcal{I}, \alpha \models q_2(\vec{x})$, for all $\mathcal{I}$ and α; or equivalently
   - $\mathcal{I}_{\alpha,\vec{c}} \models q_1(\vec{c})$ implies $\mathcal{I}_{\alpha,\vec{c}} \models q_2(\vec{c})$, for all $\mathcal{I}_{\alpha,\vec{c}}$, where $\vec{c}$ are new constants, and $\mathcal{I}_{\alpha,\vec{c}}$ extends $\mathcal{I}$ to the new constants with $c^{\mathcal{I}_{\alpha,\vec{c}}} = \alpha(x)$.
2. Construct the canonical interpretation $\mathcal{I}_{q_1(\vec{c})}$ of the CQ $q_1(\vec{c})$ on the left hand side ...
3. ...and evaluate on $\mathcal{I}_{q_1(\vec{c})}$ the CQ $q_2(\vec{c})$ on the right hand side, i.e., check whether $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.

Reducing containment of CQs to CQ evaluation

Theorem ([CM77])

For CQs, $q_1(\vec{x}) \subseteq q_2(\vec{x})$ iff $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$, where $\vec{c}$ are new constants.

Proof.

"$\Rightarrow$" Assume that $q_1(\vec{x}) \subseteq q_2(\vec{x})$.

- Since $\mathcal{I}_{q_1(\vec{c})} \models q_1(\vec{c})$ it follows that $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.

"$\Leftarrow$" Assume that $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.

- By [CM77] on hom., for every $\mathcal{I}$ such that $\mathcal{I} \models q_1(\vec{c})$ there exists a homomorphism h from $\mathcal{I}_{q_1(\vec{c})}$ to $\mathcal{I}$.
- On the other hand, since $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$, again by [CM77] on hom., there exists a homomorphism h' from $\mathcal{I}_{q_2(\vec{c})}$ to $\mathcal{I}_{q_1(\vec{c})}$.
- The mapping $h \circ h'$ (obtained by composing h and h') is a homomorphism from $\mathcal{I}_{q_2(\vec{c})}$ to $\mathcal{I}$. Hence, once again by [CM77] on hom., $\mathcal{I} \models q_2(\vec{c})$.

So we can conclude that $q_1(\vec{c}) \subseteq q_2(\vec{c})$, and hence $q_1(\vec{x}) \subseteq q_2(\vec{x})$. $\square$

Query containment for CQs

For CQs, we also have that (boolean) query evaluation $\mathcal{I} \models q$ can be reduced to query containment.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \ldots, c^{\mathcal{I}}, \ldots)$.
We construct the (boolean) CQ $q_{\mathcal{I}}$ as follows:

- $q_{\mathcal{I}}$ has no existential variables (hence no variables at all);
- the constants in $q_{\mathcal{I}}$ are the elements of $\Delta^{\mathcal{I}}$;
- for each relation P interpreted in $\mathcal{I}$ and for each fact $(a_1, \ldots, a_k) \in P^{\mathcal{I}}$, $q_{\mathcal{I}}$ contains one atom $P(a_1, \ldots, a_k)$ (note that each $a_i \in \Delta^{\mathcal{I}}$ is a constant in $q_{\mathcal{I}}$).

Theorem

For CQs, $\mathcal{I} \models q$ iff $q_{\mathcal{I}} \subseteq q$.

Query containment for CQs – Complexity

From the previous results and NP-completenss of combined complexity of CQ evaluation, we immediately get:

Theorem

Containment of CQs is NP-complete.

Query containment for CQs – Complexity

From the previous results and NP-completenss of combined complexity of CQ evaluation, we immediately get:

Theorem

Containment of CQs is NP-complete.

Since CQ evaluation is NP-complete even in query complexity, the above result can be strengthened:

Theorem

Containment $q_1(\vec{x}) \subseteq q_2(\vec{x})$ of CQs is NP-complete, even when q_1 is considered fixed.

Union of conjunctive queries (UCQs)

Def.: A union of conjunctive queries (UCQ) is a FOL query of the form

$$\bigvee_{i=1,\ldots,n} \exists \vec{y}_i . conj_i(\vec{x}, \vec{y}_i)$$

where each $conj_i(\vec{x}, \vec{y}_i)$ is a conjunction of atoms and equalities with free variables $\vec{x}$ and $\vec{y}_i$, and possibly constants.

Note: Obviously, each conjunctive query is also a of union of conjunctive queries.

Datalog notation for UCQs

A union of conjunctive queries

$$q = \bigvee_{i=1,\ldots,n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$$

is written in datalog notation as

$$\begin{array}{lccl} \{ & q(\vec{x}) & \leftarrow & conj'_1(\vec{x}, \vec{y_1}') \\ & & \vdots & \\ & q(\vec{x}) & \leftarrow & conj'_n(\vec{x}, \vec{y_n}') \quad \} \end{array}$$

where each element of the set is the datalog expression corresponding to the conjunctive query $q_i = \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$.

Note: in general, we omit the set brackets.

Evaluation of UCQs

From the definition "$\vee$" in FOL we have that:

$$\mathcal{I}, \alpha \models \bigvee_{i=1,\ldots,n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$$

if and only if

$$\mathcal{I}, \alpha \models \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i}) \qquad \text{for some } i \in \{1, \ldots, n\}.$$

Hence to evaluate a UCQ q, we simply evaluate a number (linear in the size of q) of conjunctive queries in isolation.

Hence, evaluating UCQs has the same complexity as evaluating CQs.

UCQ evaluation – Combined, data, and query complexity

Theorem (Combined complexity of UCQ evaluation)

$\{\langle \mathcal{I}, \alpha, q\rangle \mid \mathcal{I}, \alpha \models q\}$ *is NP-complete.*

- *time: exponential*
- *space: polynomial*

Theorem (Data complexity of UCQ evaluation)

$\{\langle \mathcal{I}, q\rangle \mid \mathcal{I}, \alpha \models q\}$ *is* LogSpace*-complete (query q fixed).*

- *time: polynomial*
- *space: logarithmic*

Theorem (Query complexity of UCQ evaluation)

$\{\langle \alpha, q\rangle \mid \mathcal{I}, \alpha \models q\}$ *is NP-complete (interpretation $\mathcal{I}$ fixed).*

- *time: exponential*
- *space: polynomial*

Query containment for UCQs

Theorem

For UCQs, $\{q_1, \ldots, q_k\} \subseteq \{q'_1, \ldots, q'_n\}$ iff for each q_i there is a q'_j such that $q_i \subseteq q'_j$.

Proof.

"$\Leftarrow$" Obvious.

"$\Rightarrow$" If the containment holds, then we have $\{q_1(\vec{c}), \ldots, q_k(\vec{c})\} \subseteq \{q'_1(\vec{c}), \ldots, q'_n(\vec{c})\}$, where $\vec{c}$ are new constants:

- Now consider $\mathcal{I}_{q_i(\vec{c})}$. We have $\mathcal{I}_{q_i(\vec{c})} \models q_i(\vec{c})$, and hence $\mathcal{I}_{q_i(\vec{c})} \models \{q_1(\vec{c}), \ldots, q_k(\vec{c})\}$.
- By the containment, we have that $\mathcal{I}_{q_i(\vec{c})} \models \{q'_1(\vec{c}), \ldots, q'_n(\vec{c})\}$. I.e., there exists a $q'_j(\vec{c})$ such that $\mathcal{I}_{q_i(\vec{c})} \models q'_j(\vec{c})$.
- Hence, by [CM77] on containment of CQs, we have $q_i \subseteq q'_j$. $\square$

Query containment for UCQs – Complexity

From the previous result, we have that we can check $\{q_1, \ldots, q_k\} \subseteq \{q'_1, \ldots, q'_n\}$ by at most $k \cdot n$ CQ containment checks.

We immediately get:

Theorem

Containment of UCQs is NP-complete.

References

[CM77] A. K. Chandra and P. M. Merlin.
Optimal implementation of conjunctive queries in relational data bases.
In *Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77)*, pages 77–90, 1977.

[KV98] P. G. Kolaitis and M. Y. Vardi.
Conjunctive-query containment and constraint satisfaction.
In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 205–213, 1998.

[Var82] M. Y. Vardi.
The complexity of relational query languages.
In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.