

## Lecture 2: Query Containment

Instructor: Paris Koutris

An important question to answer when we study any query language is how easy it is to decide whether two queries that are syntactically different express the same query semantically.

**Definition 2.1** (Query Equivalence). *Two queries  $q_1, q_2$  are equivalent, denoted  $q_1 \equiv q_2$ , if for every database instance  $I$ , we have  $q_1(I) = q_2(I)$ .*

**Definition 2.2** (Query Containment). *We say that query  $q_1$  is contained in query  $q_2$ , denoted  $q_1 \subseteq q_2$ , if for every database instance  $I$ , we have  $q_1(I) \subseteq q_2(I)$ .*

It is straightforward to see that if  $q_1 \subseteq q_2$  and  $q_2 \subseteq q_1$ , then  $q_1 \equiv q_2$ . Thus, we can use query containment to decide query equivalence.

In the case where both  $q_1, q_2$  are boolean queries, query containment is equivalent to *logical implication*. Indeed if  $q_1 \subseteq q_2$ , then for any instance  $I$ ,  $q_1(I) \subseteq q_2(I)$ . But since both queries are boolean, this happens if  $q_1(I)$  is false ( $\{\}$ ), or when  $q_1(I)$  is true  $\{\}$  and  $q_2(I)$  is true.

**Example 2.3.** *We will use the following two queries as our running example.*

$$q_1(x, y) : \neg R(x, y), S(y, y), R(y, w). \quad q'_1(x, y) : \neg R(x, y), S(y, z), R(z, w).$$

Is there a procedure that allows us to check whether a CQ is contained in another CQ? The answer to this question was provided by Chandra and Merlin [CM77]. But first, we need to introduce some more concepts.

**Definition 2.4** (Canonical Database). *Given a conjunctive query  $q$ , the canonical database  $D[q]$  is the database instance where each atom in the body of  $q$  becomes a fact in the instance.*

For example, the canonical database for query  $q_1$  of the running example is the instance  $D[q_1] = \{R(x, y), S(y, y), R(y, w)\}$ .

**Definition 2.5** (Homomorphism). *A homomorphism  $h$  from CQ  $q_2$  to  $q_1$  is a function  $h : \mathbf{var}(q_2) \rightarrow \mathbf{var}(q_1) \cup \mathbf{const}(q_1)$  such that:*

1. *for every atom  $R(x_1, x_2, \dots)$  in  $q_2$ , there is an atom  $R(h(x_1), h(x_2), \dots)$  in  $q_1$ .*
2.  *$h(\mathbf{head}(q_2)) = \mathbf{head}(q_1)$ , where **head** denotes the head variables of a query.*

Another term used for a homomorphism is *containment mapping*.

**Example 2.6.** *Consider the queries  $q_1, q'_1$  from the running example. Consider the function  $h$  such that  $h(x) = x, h(y) = y, h(z) = y, h(w) = w$ , and observe that it is a homomorphism from  $q'_1$  to  $q_1$ .*

## 2.1 The Homomorphism Theorem

We can now state the central theorem for query containment.

**Theorem 2.7** ([CM77]). *Given two conjunctive queries  $q_1, q_2$ , the following statements are equivalent:*

1.  $q_1 \subseteq q_2$ .
2. There exists a homomorphism  $h$  from  $q_2$  to  $q_1$ .
3.  $\mathbf{head}(q_1) \in q_2(D[q_1])$ .

*Proof.* 2  $\implies$  1. Let  $h$  be a homomorphism from  $q_2$  to  $q_1$ . Consider a database instance  $I$ . For a tuple  $t \in q_1(I)$ , we need to prove that  $t \in q_2(I)$ . Since  $t \in q_1(I)$ , there exists a valuation  $v$  such that  $t = v(\mathbf{head}(q_1))$ . Consider the composition of  $v$  with the homomorphism  $h$ :  $g = v \circ h$ . We will show that  $g$  is a valuation for query  $q_2$ . Indeed, consider any atom  $R(y_1, \dots, y_m)$  in  $q_2$ . By the definition of homomorphism,  $R(h(y_1), \dots, h(y_m))$  is an atom in the body of  $q_1$ . Then, by definition of the valuation,  $R(g(y_1), \dots, g(y_m))$  is a fact of  $I$ . Finally, observe that  $g(\mathbf{head}(q_2)) = v(h(\mathbf{head}(q_2))) = v(\mathbf{head}(q_1)) = t$ .

1  $\implies$  3. Let  $q_1 \subseteq q_2$ , and consider the canonical database  $D[q_1]$ . Then,  $\mathbf{head}(q_1) \in q_1(D[q_1])$ , since the identity function is a valuation for the canonical database. Since  $q_1 \subseteq q_2$ ,  $\mathbf{head}(q_1) \in q_2(D[q_1])$  as well.

3  $\implies$  2. Since  $\mathbf{head}(q_1) \in q_2(D[q_1])$ , there exists a valuation  $v$  for query  $q_2$ . This valuation is a homomorphism from  $q_2$  to  $q_1$ , since (a)  $v(\mathbf{head}(q_2)) = \mathbf{head}(q_1)$  and (b) for every atom  $R(y_1, \dots, y_m)$ ,  $R(v(y_1), \dots, v(y_m))$  is a fact in the canonical database  $D[q_1]$ , and so an atom in  $q_1$ .  $\square$

**Example 2.8.** *Since there exists a homomorphism from  $q'_1$  to  $q_1$ , the theorem tells us that  $q_1 \subseteq q'_1$ . On the other hand,  $q'_1$  is not contained in  $q_1$ .*

**Exercise 2.9.** *Decide whether  $q \subseteq q'$  or  $q' \subseteq q$  for the following pairs of CQs:*

$$\begin{array}{ll}
 q_2(x) : -R(x, y), R(y, z), R(z, w) & q'_2(x) : -R(x, y), R(y, z) \\
 q_3(x) : -R(x, y), R(y, z), R(z, x) & q'_3(x) : -R(x, y), R(y, x) \\
 q_4(x) : -R(x, u), R(u, u) & q'_4(x) : -R(x, u), R(u, v), R(v, w) \\
 q_5(x) : -R(x, y), R(y, z), R(z, w) & q_2(x) : -R(x, y), R(y, x)
 \end{array}$$

## 2.2 The Complexity of CQ Containment

We will now characterize the complexity of deciding whether a CQ is contained in another CQ.

**Theorem 2.10** ([CM77]). *The problem of query containment for conjunctive queries is NP-complete.*

*Proof.* The membership in  $NP$  follows from the homomorphism theorem. Indeed, to decide whether  $q_1 \subseteq q_2$ , we need to find a homomorphism from  $q_2$  to  $q_1$ . If we guess a valuation for the variables of  $q_2$ , we can check whether it is a homomorphism in polynomial time.

To show that query containment is  $NP$ -hard, we will use a reduction from the graph 3-colorability problem: *given a (directed or undirected) graph  $G(V, E)$ , is it possible to color the vertices with 3 colors, such that every pair of neighboring vertices has different colors?* Given the graph  $G$ , consider the binary relation  $R(x, y)$  that represents the edges of the graph, and let  $q_G$  be the boolean conjunctive query that corresponds to the graph. For example, if the graph has the edges  $(a, b), (b, c), (c, a)$ , then the query is  $q_G() : \neg R(a, b), R(b, c), R(c, a)$ . Consider now the query

$$K_3() : \neg R(x, y), R(y, x), R(y, z), R(z, y), R(z, x), R(x, z).$$

$K_3$  essentially represents the complete (undirected) graph with 3 vertices. It is easy to see that  $G$  is 3-colorable if and only if there exists a homomorphism from  $q_G$  to  $K_3$ . Indeed, a homomorphism will map the variables of  $q_G$ , which are the vertices of the graph  $G$ , to the variables  $x, y, z$  of  $K_3$ , which can be viewed as the 3 colors. Applying the homomorphism theorem,  $G$  is 3-colorable if and only if  $K_3 \subseteq q_G$ .  $\square$

It can also be shown (the proof is left as an exercise) that deciding the equivalence of conjunctive queries is also  $NP$ -complete.

**Theorem 2.11** ([CM77]). *The problem of query equivalence for conjunctive queries is  $NP$ -complete.*

As a corollary of the homomorphism theorem, we can also pinpoint the complexity of evaluating a conjunctive query.

**Corollary 2.12.** *The problem of evaluating a conjunctive query is  $NP$ -complete (combined complexity).*

Even though query containment is an  $NP$ -complete problem, it is a feasible problem to solve because typically the size of the queries we want to check for containment is not very large.

## 2.3 Conjunctive Query Minimization

In this section, we discuss how to minimize a given conjunctive query. In other words, given a conjunctive query  $q$ , can we find an equivalent CQ  $q'$  such that it has as few atoms as possible?

**Definition 2.13** (Minimal Query). *A conjunctive query  $q$  is minimal if for every other conjunctive query  $q'$  such that  $q \equiv q'$ ,  $q'$  has at least as many atoms as  $q$ .*

**Example 2.14.** *Consider the conjunctive query  $q(x) : \neg R(x, y), R(x, z), R(z, w)$ . This query is not minimal; indeed, consider the conjunctive query  $q'(x) : \neg R(x, z), R(z, w)$ . It is easy to see that  $q \equiv q'$ , and  $q'$  has one fewer atom than  $q$ . In contrast,  $q'$  is a minimal query.*

**Exercise 2.15.** *Find the minimal equivalent CQ to  $q(x, y) : \neg R(y, x), R(z, x), R(w, x), R(x, u)$ .*

Using query minimization to evaluate CQs sounds like a promising approach. Instead of evaluating a CQ  $q$  directly, we can first compute a minimal equivalent CQ  $q'$ , and then evaluate the new query  $q'$  to obtain the desired answer faster (since there are fewer atoms in the body, we expect that we need less computation to compute the same result).

The following theorem characterizes the structure of minimal conjunctive queries.

**Theorem 2.16.** *Let  $q$  be a conjunctive query.*

1. *There exists a minimal equivalent conjunctive query  $q'$  that can be obtained from  $q$  by removing zero or more atoms.*
2. *All minimal equivalent queries of  $q$  are isomorphic (the same up to variable renaming).*

*Proof.* (1) Let  $q''$  be a minimal equivalent query to  $q$ . Then, by the homomorphism theorem, there exists a homomorphism  $h$  from  $q$  to  $q''$ , and  $g$  from  $q''$  to  $q$ . Let  $q'$  be the query that results if we apply  $h \circ g$  to  $q$ . It is straightforward to verify that  $q' \equiv q$  and that  $q'$  has at most as many atoms as  $q''$ .

(2) This item tells us that, even though there is not a unique minimal equivalent query, minimal queries are the same up to renaming of variables. The proof is left as an exercise!  $\square$

Using the above theorem as a guide, we can devise a simple algorithm that computes a minimal equivalent query:

1. Choose an atom from  $q$  and remove it to obtain a new query  $q'$ . We know from the homomorphism theorem that  $q \subseteq q'$ .
2. Check if  $q' \subseteq q$ ; if so, then  $q'$  is equivalent and we can continue the process of removing another atom.
3. If not, try to remove another atom from  $q$ .

Unfortunately, CQ minimization is also an *NP*-hard problem, so we cannot hope to have a general efficient algorithm to minimize a given query. CQ minimization techniques are not used in practice inside query optimizers.

## 2.4 Beyond Conjunctive Queries

Query containment for all of relational algebra (so the first-order fragment of relational calculus) is actually an undecidable problem. To prove undecidability, we can reduce from the problem of finite satisfiability of first-order formulas. A first order sentence  $\phi$  is *finitely satisfiable* if there exists a finite database instance  $I$  such that  $\phi$  is true over  $I$ .

**Theorem 2.17** (Trakhtenbrot's Theorem [T50]). *Finite satisfiability is undecidable in first-order logic.*

To show the reduction, consider a first-order sentence  $\phi$  and construct the following two relational calculus queries:  $q_1 = \exists x(R(x) \wedge \phi)$  and  $q_2 = \exists x(R(x) \wedge (x \neq x))$ . It is clear that  $q_2$  is always false, hence  $q_1 \subseteq q_2$  if and only if  $q_1$  is always false, which is equivalent to  $\phi$  being not finitely satisfiable.

However, query containment is more tractable for classes of queries that are between CQs and the full relational calculus. Let's consider first the class of UCQs.

**Theorem 2.18** ([SY80]). *Let  $q = q_1 \cup q_2 \cup \dots \cup q_m$  and  $q' = q'_1 \cup q'_2 \cup \dots \cup q'_n$  be UCQs. The following statements are equivalent:*

1.  $q \subseteq q'$ .
2. For every  $i = 1, \dots, m$ , there exists some  $j = 1, \dots, n$  such that  $q_i \subseteq q'_j$ .

*Proof.* 2  $\implies$  1. Straightforward. Consider an instance  $I$  and let  $t \in q(I)$ . Then,  $t \in q_i(I)$  for some  $i = 1, \dots, m$ , and since  $q_i \subseteq q'_j$  for some  $j$ ,  $t \in q'_j(I)$ . Thus,  $t \in q'(I)$ .

1  $\implies$  2. Consider the canonical instance  $D[q_i]$ . We then have:  $q_i(D[q_i]) \subseteq q(D[q_i]) \subseteq q'(D[q_i])$ . It must be then that  $\text{head}(q_i) \in q'_j(D[q_i])$ , which by the homomorphism theorem implies that  $q_i \subseteq q'_j$ .  $\square$

As a corollary of the above result, we obtain that the problem of query containment for unions of conjunctive queries is also an *NP*-complete problem.

The complexity for query containment for  $CQ^\neq, CQ^<$  is much higher than *NP*-complete: it is in the complexity class  $\Pi_2^P$ .

## References

- [Alice] S. ABITEBOUL, R. HULL and V. VIANU, "Foundations of Databases."
- [CM77] A.K. CHANDRA and P.M. MERLIN, "Optimal implementation of conjunctive queries in relational data bases," *STOC 1977*.
- [SY80] Y. SAGIV and M. YANNAKAKIS, "Equivalences Among Relational Expressions with the Union and Difference Operators", *JACM 1980*.
- [T50] B.A. TRAKHTENBROT, "The impossibility of an algorithm for the decidability problem on finite classes", *Doklady AN SSR 70(4)*, 569?572, 1950.