In this lecture we will study when we can say that two conjunctive queries that are syntactically different express the same query, or when the output of one query is always contained in the output of another query.

**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 $q_1 \equiv q_2$. Notice that when both queries are boolean queries, query containment becomes equivalent to logical implication!

**Exercise 2.3.** Decide whether $q \subseteq q'$ for the following pairs of CQs:

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

$$
q_2(x) : -R(x, y), R(y, z), R(z, w) \quad q'_2(x) : -R(x, y), R(y, z)
$$

$$
q_3(x) : -R(x, y), R(y, z), R(z, x) \quad q'_3(x) : -R(x, y), R(y, x)
$$

$$
q_4(x) : -R(x, u), R(u, u) \quad q'_4(x) : -R(x, u), R(u, v), R(v, w)
$$

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 $q$ becomes a fact in the instance.

As an 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 $q_2$ to $q_1$ is a function $h : \text{var}(q_2) \rightarrow \text{var}(q_1) \cup \text{const}(q_1)$ such that:

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

Another term used for such 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
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. \( \text{head}(q_1) \in q_2(D[q_1]) \).

**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 \( t = v(\text{head}(q_1)) \). Consider the composition of \( v \) with the homomorphism \( h: g = v \circ h \). We will show that \( g \) is a valuation for query \( q_2 \). Consider now an atom \( R(y_1, \ldots, y_m) \) in \( q_2 \). By the definition of homomorphism, \( R(h(y_1), \ldots, h(y_m)) \) is in \( q_1 \), and then by definition of the valuation, \( R(g(y_1), \ldots, g(y_m)) \) is a fact of \( I \). Also, \( g(\text{head}(q_2)) = v(h(\text{head}(q_2))) = v(\text{head}(q_1)) = t \).

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

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

**Exercise 2.8.** Let \( q_1(x) : -R(x, y), R(y, z), R(z, w) \) and \( q_2(x) : -R(x, y), R(y, z) \). Is \( q_2 \subseteq q_1 \)?

**Theorem 2.9 ([CM77]).** The problem of query containment for conjunctive queries is \( \text{NP-complete} \).

**Proof.** The \( \text{NP} \)-hardness is proved by reduction from graph 3-colorability: given a 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 \( (a, b), (b, c), (c, d), (c, a) \), then the query is \( q_G() : -R(a, b), R(b, c), R(c, d), R(c, a) \). Consider now the query \( K_3() : -R(x, y), R(y, z), R(z, x) \). It is easy to see that \( G \) is 3-colorable if and only if there exists a homomorphism from \( q_G \) to \( K_3 \). By the homomorphism theorem, \( G \) is 3-colorable if and only if \( K_3 \subseteq q_G \).

The membership in \( \text{NP} \) follows again from the homomorphism theorem. To decide whether \( q_1 \subseteq q_2 \), we need to find a homomorphism from \( q_2 \) to \( q_1 \). If we guess a function, we can check whether it is a homomorphism in polynomial time.

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

**Corollary 2.10.** The problem of evaluating a conjunctive query is \( \text{NP-complete} \).
It can be shown that query equivalence for conjunctive queries is also \( \text{NP} \)-complete. Even though query containment is an \( \text{NP} \)-complete problem, it is feasible problem to solve because typically the size of the query is not very large.

### 2.2 Query Minimization

Here we discuss how we can 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 2.11 (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 \( q \).

**Exercise 2.12.** Is the query \( q(x) : \neg R(x, y), R(x, z), R(z, w) \) minimal? What about the query \( q(x) : \neg R(x, a), R(x, y), R(y, z) \)?

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.

**Theorem 2.13.** 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.

**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!  

Below is a simple procedure to compute a minimal equivalent query:

- Choose an atom from \( q \) and remove it to obtain a new query \( q' \). We know from the homomorphism theorem that \( q \subseteq q' \).
- Check if \( q' \subseteq q \); if so, then \( q' \) is equivalent and we can continue the process of removing another atom.
- If not, try to remove another atom from \( q \).
Unfortunately, CQ minimization is an \emph{NP}-hard problem, so we cannot hope to have a general efficient algorithm. Even though CQ minimization looks like a promising avenue, in practice query optimizers do not use it (why?).

### 2.3 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 \emph{finitely satisfiable} if there exists a finite database instance $I$ such that $\phi$ is true over $I$.

\textbf{Theorem 2.14} (Trakhtenbrot’s Theorem [T50]). \emph{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:

$$q_1 = \exists x (R(x) \land \phi) \text{ and } q_2 = \exists x (R(x) \land (x \neq x)).$$

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, as we will see next, 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.

\textbf{Theorem 2.15} ([SY80]). Let $q = q_1 \cup q_2 \cup \cdots \cup q_m$ and $q' = q'_1 \cup q'_2 \cup \cdots \cup q'_n$ be UCQs. The following statements are equivalent:

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

\textbf{Proof.} $2 \implies 1$. Straightforward. Consider an instance $I$ and let $t \in q(I)$. Then, $t \in q_i(I)$ for some $i = 1, \ldots, m$, 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 $q_i \subseteq q'_j$.

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

The complexity for query containment for $CQ^\neq, CQ^\prec$ is much higher than \emph{NP}-complete: it is in the class $\Pi^p_2$. We won’t get into details for this complexity result.

\section*{References}

[Alice] S. \textsc{Abiteboul}, R. \textsc{Hull} and V. \textsc{Vianu}, “Foundations of Databases.”
