The Homomorphism Problem and Conjunctive Queries

- **Theorem**: Chandra & Merlin, 1977
  - CQE and CQC are the “same” problem.
  - Moreover, each is an NP-complete problem.

- **Question**: What is the common link?
- **Answer**:
  - Both CQE and CQC are “equivalent” to the Homomorphism Problem.
  - The link is established by bringing into the picture
    - Canonical conjunctive queries and
    - Canonical database instances.
Homomorphisms, CQE, and CQC

**The Homomorphism Theorem:** Chandra & Merlin – 1977
For Boolean CQs Q and Q’, the following are equivalent:
- \( Q \subseteq Q' \)
- There is a homomorphism \( h: I^Q \rightarrow I^Q \)
- \( I^Q \models Q' \).

In dual form:

**The Homomorphism Theorem:** Chandra & Merlin – 1977
For instances I and I’, the following are equivalent:
- There is a homomorphism \( h: I \rightarrow I' \)
- \( I' \models Q^I \)
- \( Q'^I \subseteq Q^I \)
The Homomorphism Theorem for non-Boolean Conjunctive Queries

**The Homomorphism Theorem:** Chandra & Merlin – 1977

Consider two k-ary conjunctive queries

\[ Q(x_1, ..., x_k) : \neg R_1(u_1), ..., R_n(u_n) \]

and

\[ Q'(x_1, ..., x_k) : \neg T_1(v_1), ..., T_m(v_m), \]

Then the following are equivalent:

- \( Q \subseteq Q' \)

- There is a homomorphism \( h: I^Q \rightarrow I^Q \) such that
  \[ h(x_1) = x_1, h(x_2) = x_2, ..., h(x_k) = x_k. \]

- \( I^Q, x_1, ..., x_k \models Q'. \)
The Homomorphism Theorem for non-Boolean Conjunctive Queries

**Example:** Consider the binary conjunctive queries

\[ Q(x,y) :\ E(y,x), E(x,u) \]

and

\[ Q'(x,y) :\ E(y,x), E(z,x), E(w,x), E(x,u) \]

Then \( Q \subseteq Q' \) because there is a homomorphism

\[ h: \mathcal{I}^{Q'} \rightarrow \mathcal{I}^{Q} \]

with \( h(x) = x \) and \( h(y) = y \),

namely,

\[ h(x) = x, h(y) = y, h(z) = y, h(w) = y, h(u) = u. \]
### The Complexity of Database Query Languages

<table>
<thead>
<tr>
<th>Query Evaluation Problem: Combined Complexity</th>
<th>Relational Calculus</th>
<th>Conjunctive Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSPACE-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Evaluation Problem: Data Complexity</th>
<th>In LOGSPACE (hence, in P)</th>
<th>In LOGSPACE (hence, in P)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Query Equivalence Problem</th>
<th>Undecidable</th>
<th>NP-complete</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Query Containment Problem</th>
<th>Undecidable</th>
<th>NP-complete</th>
</tr>
</thead>
</table>
Beyond Conjunctive Queries

- What can we say about query languages of intermediate expressive power between conjunctive queries and the full relational calculus?

- Conjunctive queries form the sublanguage of relational algebra obtained by using only cartesian product, projection, and selection with equality conditions.

- The next step would be to consider relational algebra expressions that also involve union.
Beyond Conjunctive Queries

■ Definition:
  - A union of conjunctive queries is a query expressible by an expression of the form \( q_1 \cup q_2 \cup ... \cup q_m \), where each \( q_i \) is a conjunctive query.
  - A monotone query is a query expressible by a relational algebra expression which uses only union, cartesian product, projection, and selection with equality condition.

■ Fact:
  - Every union of conjunctive queries is a monotone query.
  - Every monotone query is equivalent to a union of conjunctive queries, but the union may have exponentially many disjuncts. (normal form for monotone queries).
  - Monotone queries are precisely the queries expressible by relational calculus expressions using \( \land, \lor, \text{ and } \exists \text{ only.} \)
Unions of Conjunctive Queries and Monotone Queries

- **Union of Conjunctive Queries**
  \[ E \cup \pi_{1,4} (\sigma_{x_2=x_3} (E \times E)) \text{ or, as a relational calculus expression,} \]
  \[ E(x_1,x_2) \lor \exists z(E(x_1,z) \land E(z,x_2)) \]

- **Monotone Query**
  Consider the relation schemas \( R_1(A,B), R_2(A,B), R_3(B,C), R_4(B,C) \).
  The monotone query
  \[ (R_1 \cup R_2) \bowtie (R_3 \cup R_4) \]
  is equivalent to the following union of conjunctive queries:
  \[ (R_1 \bowtie R_3) \cup (R_1 \bowtie R_4) \cup (R_2 \bowtie R_3) \cup (R_2 \bowtie R_4). \]
The Containment Problem for Unions of Conjunctive Queries

Theorem: Sagiv and Yannakakis – 1981
Let \( q_1 \cup q_2 \cup \ldots \cup q_m \) and \( q'_1 \cup q'_2 \cup \ldots \cup q'_n \) be two unions of conjunctive queries. Then the following are equivalent:
1. \( q_1 \cup q_2 \cup \ldots \cup q_m \subseteq q'_1 \cup q'_2 \cup \ldots \cup q'_n \).
2. For every \( i \leq m \), there is \( j \leq n \) such that \( q_i \subseteq q'_j \).

Proof: Use the Homomorphism Theorem
1. \( \Rightarrow \) 2. Since \( I^q_i \models q_i \), we have that \( I^q_i \models q_1 \cup q_2 \cup \ldots \cup q_m \), hence \( I^q_i \models q'_1 \cup q'_2 \cup \ldots \cup q'_n \), hence there is some \( j \leq n \) such that \( I^q_i \models q'_j \), hence (by the Homomorphism Theorem) \( q_i \subseteq q'_j \).

2. \( \Rightarrow \) 1. This direction is obvious.
### The Complexity of Database Query Languages

<table>
<thead>
<tr>
<th></th>
<th>Relational Calculus</th>
<th>Conjunctive Queries</th>
<th>Unions of Conjunctive Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query Evaluation: Combined Complexity</strong></td>
<td>PSPACE-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td><strong>Query Evaluation: Data Complexity</strong></td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
</tr>
<tr>
<td><strong>Query Equivalence</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td><strong>Query Containment</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
</tbody>
</table>
Monotone Queries

- Even though monotone queries have the same expressive power as unions of conjunctive queries, the containment problem for monotone queries has higher complexity than the containment problem for unions of conjunctive queries (syntax/complexity tradeoff).

- **Theorem:** Sagiv and Yannakakis – 1982
  The containment problem for monotone queries is $\Pi_2^p$-complete.

- **Note:**
  - $\Pi_2^p$ is a complexity class that contains NP and is contained in PSPACE.
  - The prototypical $\Pi_2^p$-complete problem is $\forall \exists$-SAT, i.e., the restriction of QBF to formulas of the form
    \[ \forall x_1 \cdots \forall x_m \exists y_1 \cdots \exists y_n \varphi. \]
Monotone Queries

Computational Complexity Classes

- LOGSPACE
- NLOGSPACE
- P
- NP
- $\Pi_2^p$
- PSPACE

Monotone Queries

<table>
<thead>
<tr>
<th>Problem</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Containment</td>
<td>$\Pi_2^p$-complete</td>
</tr>
<tr>
<td>Query Equivalence</td>
<td>$\Pi_2^p$-complete</td>
</tr>
<tr>
<td>Query Evaluation</td>
<td>NP-complete</td>
</tr>
<tr>
<td>(combined complexity)</td>
<td></td>
</tr>
</tbody>
</table>
Computational Complexity Classes

Facts:
- \( P \subseteq NP \subseteq \Pi^P_2 \subseteq PH \subseteq PSPACE \)
  (these containments are believed to be proper)
- PH is the polynomial hierarchy, which is obtained by starting with NP and then considering all decision problems solvable by a (deterministic) polynomial-time Turing machine with oracles for solving decision problems from lower levels of PH.
- \( \Pi^P_2 \) is the second level of PH.
- \( NP \cup coNP \subseteq \Pi^P_2 \).
- For more information, visit the Petting Complexity Zoo
  http://qwiki.stanford.edu/wiki/Petting_Zoo
## The Complexity of Database Query Language

<table>
<thead>
<tr>
<th></th>
<th>Relational Calculus</th>
<th>Conjunctive Queries</th>
<th>Unions of Conjunctive Queries</th>
<th>Monotone Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query Eval.: Combined Complexity</strong></td>
<td>PSPACE-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td><strong>Query Eval.: Data Complexity</strong></td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
</tr>
<tr>
<td><strong>Query Equivalence</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>$\Pi_2^p$-complete</td>
</tr>
<tr>
<td><strong>Query Containment</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>$\Pi_2^p$-complete</td>
</tr>
</tbody>
</table>
Conjunctive Queries with Inequalities

- **Definition:** Conjunctive queries with inequalities form the sublanguage of relational algebra obtained by using only cartesian product, projection, and selection with equality and inequality (≠, <, ≤) conditions.

- **Example:** Relational schema Movie(title, star)
  - Find all movies starring at least two different governors.
  - Q(x) :- Movie(x,y), Movie(x,z), Governor(y), Governor(z), y ≠ z

- **Example:** Q(x,y):-- E(x,z), E(z,w), E(w,y), z ≠ w, z < y.

- **Note:** The use of < assumes a total order on the domain of values.
Conjunctive Queries with Inequalities

- FACULTY(name, dpt, salary), CHAIR(name, dpt)

- Find the names of faculty who earn a higher salary than that of their department chair:

  \[Q(x) : \text{FACULTY}(x,y,z), \text{FACULTY}(x',y,z'), \text{CHAIR}(x',y), z > z'\]

- Find the names of faculty who are not the lowest paid ones in their department

  \[Q(x) : \text{FACULTY}(x,y,z), \text{FACULTY}(x',y,z'), z > z'\]
Conjunctive Queries with Inequalities

**Theorem:** The query containment problem for conjunctive queries with ≠, ≤, ≥ is \( \Pi_2^p \)-complete.

- **A. Klug** – 1988: Membership in \( \Pi_2^p \).
  It suffices to test containment on exponentially many “canonical” databases, each of size polynomial in the size of the queries.

- **R. van der Meyden** – 1992: \( \Pi_2^p \)-hardness
  \( \Pi_2^p \)-hardness holds, even for conjunctive queries with only ≠.
  (difficult reduction)
Note: The Homomorphism Theorem fails for conjunctive queries with inequalities.

Example: Consider the queries

- \( q(x,y) \) :- \( E(x,y), F(u,v) \)
- \( p(x,y) \) :- \( E(x,y), F(u,v), u \neq v \).

Then

- \( q(x,y) \not\subseteq p(x,y) \) (why?)
- Yet, \( I_p \) is the same as \( I_q \);
  - in particular, there is a homomorphism \( h: I_p \rightarrow I_q \).
- Note also that \( p(x,y) \subseteq q(x,y) \) (why?)
## The Complexity of Database Query Languages

<table>
<thead>
<tr>
<th></th>
<th>Relational Calculus</th>
<th>Conjunctive Queries</th>
<th>Unions of Conjunctive Queries</th>
<th>Monotone Queries/ Conj. Queries with ≠</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query Eval.: Combined Complexity</strong></td>
<td>PSPACE-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>NP-complete</td>
</tr>
<tr>
<td><strong>Query Eval.: Data Complexity</strong></td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
<td>In LOGSPACE (hence, in P)</td>
</tr>
<tr>
<td><strong>Query Equivalence</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>$II_2^p$-complete</td>
</tr>
<tr>
<td><strong>Query Containment</strong></td>
<td>Undecidable</td>
<td>NP-complete</td>
<td>NP-complete</td>
<td>$II_2^p$-complete</td>
</tr>
</tbody>
</table>
Complexity of Query Containment

- So, the complexity of query containment for conjunctive queries and their variants is well understood.

**Caveat:**
- All preceding results assume set semantics, i.e., queries take sets as inputs and return sets as output (duplicates are eliminated).
- DBMS, however, use bag semantics (multiset semantics), since they return bags (multisets) (recall that duplicates are not eliminated in SQL, unless explicitly specified).
A *Real* Conjunctive Query

- Consider the following SQL query:
  Table Employee has attributes salary, dept, ...
  ```sql
  SELECT salary
  FROM Employee
  WHERE dept = 'CS'
  ```

- Recall that SQL keeps duplicates, because:
  - User may care about duplicates
    - \{100, 100, 200\} different than \{100, 200\} for **AVERAGE**
  - In general, bags can be more “efficient” than sets.
## Query Evaluation under Bag Semantics

<table>
<thead>
<tr>
<th>Operation</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>$m_1 + m_2$</td>
</tr>
<tr>
<td>$R_1 \cup R_2$</td>
<td></td>
</tr>
<tr>
<td>Intersection</td>
<td>$\min(m_1, m_2)$</td>
</tr>
<tr>
<td>$R_1 \cap R_2$</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>$m_1 \times m_2$</td>
</tr>
<tr>
<td>$R_1 \times R_2$</td>
<td></td>
</tr>
<tr>
<td>Projection and Selection</td>
<td>Duplicates are not eliminated</td>
</tr>
</tbody>
</table>

- **$R_1$**
  
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- **$R_2$**
  
<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

- **$(R_1 \bowtie R_2)$**
  
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Bag (Multiset) Semantics

S. Chaudhuri & M.Y. Vardi – 1993

Optimization of *Real* Conjunctive Queries

- Called for a re-examination of conjunctive-query optimization under bag semantics.

- In particular, they initiated the study of the containment problem for conjunctive queries under bag semantics.
Bag Semantics vs. Set Semantics

- $Q^{\text{BAG}}(I)$: Result of evaluating $Q$ on (bag) database instance $I$.
- For bags $R_1$, $R_2$:
  
  $R_1 \subseteq_{\text{BAG}} R_2$ if $m(a, R_1) \leq m(a, R_2)$, for every tuple $a$.
- $Q_1 \subseteq_{\text{BAG}} Q_2$ if for every (bag) database $I$, we have that $Q_1^{\text{BAG}}(I) \subseteq_{\text{BAG}} Q_2^{\text{BAG}}(I)$.

**Fact:**

- $Q_1 \subseteq_{\text{BAG}} Q_2$ implies $Q_1 \subseteq Q_2$ (this is obvious from the definitions).
- The converse does **not** always hold.
Bag Semantics vs. Set Semantics

Fact:
- $Q_1 \subseteq_{BAG} Q_2$ implies $Q_1 \subseteq Q_2$ (this is obvious from the definitions).
- The converse does **not** always hold:
  - $Q_1(x,y) :\neg P(x,y), R(x,z)$
  - $Q_2(x,y) :\neg P(x,y)$
  Then $Q_1 \subseteq Q_2$, but **not** $Q_1 \subseteq_{BAG} Q_2$ (Why?)

Fact: The following statements need **not** be equivalent:
- $Q_1 \subseteq_{BAG} Q_2$
- $Q_1 \land Q_2 \equiv_{BAG} Q_1$
Conjunctive Query Evaluation under Bag Semantics

Consider:

- $K_3$: the complete graph with 3 nodes
- $G=(V,E)$ an arbitrary graph and the canonical conjunctive query $Q^G$ of $G$.

Then

- **Set Semantics**
  
  \[ Q^G(K_3) = \text{“true” if and only if G is 3-colorable.} \]

- **Bag Semantics**
  
  \[ Q^{G\text{-BAG}}(K_3) = \# \text{3-colorings of the graph G}. \]

**Corollary:** The conjunctive query evaluation problem under bag semantics is $\#P$-complete.
Chaudhuri & Vardi - 1993 stated that:
Under bag semantics, the containment problem for conjunctive queries is $\Pi_2^p$-hard.

**Open Problem:**
- What is the exact complexity, under bag semantics, of the containment problem for conjunctive queries?
- Is this problem decidable? Even this is not known to date!
Unions of Conjunctive Queries

**Theorem:** Ioannidis & Ramakrishnan – 1995
Under bag semantics, the containment problem for unions of conjunctive queries is **undecidable**.

**Hint of Proof:**
Reduction from Hilbert’s 10th Problem.
Hilbert’s 10\textsuperscript{th} Problem

- Hilbert’s 10\textsuperscript{th} Problem – 1900
  (10\textsuperscript{th} in his list of 23 problems)
  Find an algorithm for the following problem:
  Given a polynomial equation $p(x_1,...,x_n) = 0$ with integer coefficients,
  does it have an all-integer solution?

- Matijasevic – 1971
  - Hilbert’s 10\textsuperscript{th} Problem is **undecidable**, hence no such algorithm
    exists.
  - **Undecidable**, even for degree $d = 4$ and $n = 58$. 


Fact: The following variant of Hilbert’s 10th Problem is undecidable:

- Given two polynomials $p_1(x_1, \ldots, x_n)$ and $p_2(x_1, \ldots, x_n)$ with positive integer coefficients and no constant terms, is it true that $p_1 \leq p_2$? i.e., is it true that $p_1(a_1, \ldots, a_n) \leq p_2(a_1, \ldots, a_n)$, for all positive integers $a_1, \ldots, a_n$?

So, there is no algorithm for deciding questions like:

- Is $3x_1^4x_2x_3 + 2x_2x_3 \leq x_1^6 + 5x_2x_3$?
Theorem: Ioannidis & Ramakrishnan – 1995
Under bag semantics, the containment problem for unions of conjunctive queries is undecidable, even if all relations are unary.

Hint of Proof:
- Reduction from the previous variant of Hilbert’s 10th Problem:
  - Use joins of unary relations to encode monomials (products of variables).
  - Use unions to encode sums of monomials.
Theorem: Ioannidis & Ramakrishnan – 1995
Under bag semantics, the containment problem for unions of conjunctive queries is **undecidable**, even if all relations are unary.

Example: Consider the polynomial $3x_1^4x_2x_3 + 2x_2x_3$

- The monomial $x_1^4x_2x_3$ is encoded by the conjunctive query $P_1(w), P_1(w), P_1(w), P_1(w), P_2(w), P_3(w)$.

- The monomial $x_2x_3$ is encoded by the conjunctive query $P_2(w), P_3(w)$.

- The polynomial $3x_1^4x_2x_3 + 2x_2x_3$ is encoded by the union having:
  - three copies of $P_1(w), P_1(w), P_1(w), P_1(w), P_2(w), P_3(w)$ and
  - two copies of $P_2(w), P_3(w)$. 

---

Unions of Conjunctive Queries
# Computational Complexity of Query Containment

<table>
<thead>
<tr>
<th>Class of Queries</th>
<th>Complexity – Set Semantics</th>
<th>Complexity – Bag Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunctive queries</td>
<td>NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>Chandra Merlin – 1977</td>
<td></td>
</tr>
<tr>
<td>Unions of conj. queries</td>
<td>NP-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Conj. queries with $\neq, \leq, \geq$</td>
<td>$\Pi^p_2$-complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Van der Meyden – 1992</td>
<td></td>
</tr>
<tr>
<td>Relational calculus queries</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Trakhtenbrot - 1949</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Variance of complexity</td>
<td></td>
</tr>
</tbody>
</table>
Bag Semantics and Conjunctive Queries with $\neq$

**Theorem**: Jayram, K... , Vee – 2006
Under bag semantics, the containment problem for conjunctive queries with $\neq$ is **undecidable**.

In fact, this problem is **undecidable** even if
- the queries use only a **single** relation of arity 2;
- the **number of inequalities** in the queries is at most some **fixed constant**.
Bag Semantics and Conjunctive Queries with ≠

**Proof Idea:**
Reduction from another variant of Hilbert’s 10th Problem:

Given homogeneous polynomials $P_1(x_1,\ldots,x_{59})$ and $P_2(x_1,\ldots,x_{59})$ both with integer coefficients and both of degree 5, is $P_1(x_1,\ldots,x_{59}) \leq (x_1)^5 P_2(x_1,\ldots,x_{59})$, for all integers $x_1,\ldots,x_{59}$?

The reduction is much more involved than the earlier reduction for unions of conjunctive queries.
Proof Idea (continued)

- Given polynomials $P_1$ and $P_2$
  - Both with integer coefficients
  - Both homogeneous, degree 5
  - Both with at most $n=59$ variables
- We want to find $Q_1$ and $Q_2$ such that
  - $Q_1$ and $Q_2$ are conjunctive queries with inequalities $\gamma_0$
  - $P_1(x_1,\ldots,x_{59}) \leq (x_1)^5 P_2(x_1,\ldots,x_{59})$
    for all integers $x_1, \ldots, x_{59}$
    if and only if
    $Q_1(D) \subseteq_{\text{BAG}} Q_2(D)$ for all (bag) databases $D$. 
Proof Outline:

Proof is carried out in three steps.

**Step 1:** Only consider DBs of a special form.  
Show how to use conjunctive queries to encode polynomials (without using inequalities!)

**Step 2:** “Force” DB to have special form using inequalities.  
- If $D$ is not of special form, then $Q_1(D) \subseteq_{BAG} Q_2(D)$ necessarily.

**Step 3:** Show that we only need a single relation of arity 2.
Step 1: DBs of a Special Form - Example

- Encode a homogeneous, 2-variable, degree 2 polynomial in which all coefficients are 1.
  \[ P(x_1,x_2) = x_1^2 + x_1x_2 + x_2^2 \]
- DBs of special form:
  - Ternary relation TERM consisting of
    - \((X_1,X_1,T_1), (X_1,X_2,T_2), (X_2,X_2,T_3)\)
    all special DBs have precisely this table for TERM
  - Binary relation VALUE
    - Table for VALUE varies to encode different values for the variables \(x_1, x_2\).
- Query \(Q : - \text{TERM}(u_1,u_2,t), \text{VALUE}(u_1,v_1), \text{VALUE}(u_2,v_2)\)
Step 1: DBs of a Special Form - Example

- \( P(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 \)
  \( x_1 = 3, \ x_2 = 2 \), \( P(3,2) = 3^2 + 3 \cdot 2 + 2^2 = 19 \).
- Query \( Q : \text{TERM}(u_1,u_2,t), \text{VALUE}(u_1,v_1), \text{VALUE}(u_2,v_2) \)
- DB \( D \) of special form:
  - \( \text{TERM}: \ (X_1,X_1,T_1), \ (X_1,X_2,T_2), \ (X_2,X_2,T_3) \)
  - \( \text{VALUE}: \ (X_1,1), \ (X_1,2), \ (X_1,3) \)
    \( (X_2,1), \ (X_2,2) \)

**Claim:** \( P(3,2) = 19 = Q^{\text{BAG}}(D) \)
Step 1: DBs of a Special Form - Example

- \( P(3,2) = 3^2 + 3 \cdot 2 + 2^2 = 19. \)
- Query \( Q : \text{TERM}(u_1, u_2, t), \text{VALUE}(u_1, v_1), \text{VALUE}(u_2, v_2) \)
- \( D \) has \( \text{TERM: } (X_1, X_1, T_1), (X_1, X_2, T_2), (X_2, X_2, T_3) \)
  \( \text{VALUE: } (X_1, 1), (X_1, 2), (X_1, 3), (X_2, 1), (X_2, 2) \)
- \( Q^{\text{BAG}}(D) = 19, \) because:
  - \( t \to T_1, u_1 \to X_1, u_2 \to X_1. \) Hence:
    - \( v_1 \to 1, 2, \) or \( 3 \) and \( v_2 \to 1, 2, \) or \( 3 \) so we get \( 3^2 \) witnesses.
  - \( t \to T_2, u_1 \to X_1, u_2 \to X_2. \) Hence:
    - \( v_1 \to 1, 2, \) or \( 3 \) and \( v_2 \to 1 \) or \( 2, \) so we get \( 3 \cdot 2 \) witnesses.
  - \( t \to T_3, u_1 \to X_2, u_2 \to X_2. \) Hence:
    - \( v_1 \to 1 \) or \( 2, \) and \( v_2 \to 1 \) or \( 2, \) so we get \( 2^2 \) witnesses.
Step 1: Complete Argument and Wrap-up

- Previous technique only works if all coefficients are 1
- For the complete argument:
  - add a fixed table for every term to the DB;
  - encode coefficients in the query;
  - only table for VALUE can vary.
- **Summary:**
  - If the database has a special form, then we can encode separately homogeneous polynomials $P_1$ and $P_2$ by conjunctive queries $Q_1$ and $Q_2$.
  - By varying table for VALUE, we vary the variable values.
  - No $\neq$-constraints are used in this encoding; hence, conjunctive query containment is **undecidable**, if restricted to databases of the special form.
Step 2: Arbitrary Databases

**Idea:**
Use inequalities ≠ in the encoding queries to achieve the following:

- If a database $D$ is of special form, then we are back to the previous case.
- If a database $D$ is not of special form, then $Q_1(D) \subseteq_{BAG} Q_2(D)$ necessarily.
Step 2: Arbitrary Databases - Hint

1. Ensure that certain “facts” in special-form DBs appear (else neither query is satisfied).
   - This is done by adding a part of the canonical query of special-form DBs as subgoals to each encoding query.

2. Modify special-form DBs by adding gadget tuples to TERM and to VALUE.
   - TERM: \((X_1, X_1, T_1), (X_1, X_2, T_2), (X_2, X_2, T_3), (T_0, T_0, T_0)\)
   - VALUE: \((X_1, 1), (X_1, 2), (X_1, 3), (X_2, 1), (X_2, 2), (T_0, T_0)\)

3. Add extra subgoals to \(Q_2\), so that if \(D\) is not of special form, then \(Q_2\) “benefits” more than \(Q_1\) and, as a result, \(Q_1(D) \subseteq_{BAG} Q_2(D)\).
Step 2: Arbitrary Databases - Example

- \( P_1(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2 \)
- \( \text{Poly}_1(u_1, u_2, t) :- \text{TERM}(u_1, u_2, t), \text{VALUE}(u_1, v_1), \text{VALUE}(u_2, v_2) \)
  the query encoding \( P_1 \) on special-form DBs.
  - \( \text{TERM}: (X_1, X_1, T_1), (X_1, X_2, T_2), (X_2, X_2, T_3), (T_0, T_0, T_0) \)
  - \( \text{VALUE}: (X_1, 1), (X_1, 2), (X_1, 3), (X_2, 1), (X_2, 2), (T_0, T_0) \)

- \( Q_1 :- \text{Poly}_1(u_1, u_2, t) \)
- \( Q_2 :- \text{Poly}_2(u_1, u_2, t), \text{Poly}_1(w_1, w_2, w), w \neq T_1, w \neq T_2, w \neq T_3 \)

**Fact:**
- If DB is of special form, then \( Q_2 \) gets no advantage, because \( w \rightarrow T_0, w_1 \rightarrow T_0, w_2 \rightarrow T_0 \) is the only possible assignment.
- If DB not of special form, say it has an extra fact \((X_2, X_1, T')\),
Step 2: Arbitrary Databases – Wrap-up

- Additional tricks are needed for the full construction.

- Full construction uses seven different control gadgets.
  - Additional complications when we encode coefficients.
  - Inequalities $\neq$ are used in both queries.

- Number of inequalities $\neq$ depends on size of special-form DBs, not counting the facts in VALUE table.
  - Hence, depends on degree of polynomials, # of variables.
  - It is a huge constant (about $59^{10}$).
## Complexity of Query Containment

<table>
<thead>
<tr>
<th>Class of Queries</th>
<th>Complexity – Set Semantics</th>
<th>Complexity – Bag Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunctive queries</td>
<td>NP-complete</td>
<td>Open</td>
</tr>
<tr>
<td></td>
<td>CM – 1977</td>
<td></td>
</tr>
<tr>
<td>Unions of conj. queries</td>
<td>NP-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>SY - 1980</td>
<td>IR - 1995</td>
</tr>
<tr>
<td>Conj. queries with ≠ , ≤ , ≥</td>
<td>$\Pi_2^p$-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>vdM - 1992</td>
<td>JKV - 2006</td>
</tr>
<tr>
<td>First-order (SQL) queries</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Gödel - 1931</td>
<td></td>
</tr>
</tbody>
</table>
Directions for Future Work

- **Major Open Problem:**
  Conjunctive query containment problem (no inequalities), under bag semantics.
  - Is it **decidable**?
  - If so, what is the exact complexity?

- Identify classes of queries for which, under bag semantics, the containment problem is **tractable**.

- Pinpoint the exact complexity of **bag-equivalence**.
Query Containment vs. Query Equivalence

- **Under set semantics,** query equivalence and query containment are reduced to each other, as long as the language is closed under conjunctions. In particular, they are of the same complexity for conjunctive queries and for conjunctive queries with inequalities ≠.

- **Under bag semantics,**
  - Query equivalence is reducible to query containment
  - Query containment need not be reducible to query equivalence.

- **Under bag semantics,** query equivalence and query containment may have very different complexity.
The Query Equivalence Problem

- **Query Equivalence**: given \( Q_1, Q_2 \), is \( Q_1 \equiv Q_2 \)?
- Under bag semantics,
  - For conjunctive queries, it has the same complexity as GRAPH ISOMORPHISM
    - Chaudhuri & Vardi - 1993
  - For conjunctive queries with inequalities \( \neq \),
    - **Lower Bound**: It is GRAPH ISOMORPHISM-hard.
    - **Upper Bound**: It is in PSPACE
      - Nutt, Sagin, Shurin (Cohen) – 1998
    Big gap between the lower and the upper bound.