Thematic Roadmap

✓ Logic and Database Query Languages
  – Relational Algebra and Relational Calculus
  – Conjunctive queries and their variants
  – Datalog

✓ Query Evaluation, Query Containment, Query Equivalence
  – Decidability and Complexity

✓ Other Aspects of Conjunctive Query Evaluation
  • Alternative Semantics of Queries
    – Bag Databases: Semantics and Conjunctive Query Containment
    – Probabilistic Databases: Semantics and Dichotomy Theorems for Conjunctive Query Evaluation
    – Inconsistent Databases: Semantics and Dichotomy Theorems
Alternative Semantics

• So far, we have examined logic and databases under classical semantics:
  – The database relations are sets.
  – Tarskian semantics are used to interpret queries definable be first-order formulas.
• Over the years, several different alternative semantics of queries have been investigated. We will discuss three such scenarios:
  – The database relations can be bags (multisets).
  – The databases may be probabilistic.
  – The databases may be inconsistent.
Sets vs. Multisets

Relation EMPLOYEE(name, dept, salary)

• Relational Algebra Expression:
  \[ \pi_{\text{salary}} (\sigma_{\text{dept} = \text{CS}} (\text{EMPLOYEE})) \]

• SQL query:
  
  ```sql
  SELECT   salary
  FROM      EMPLOYEE
  WHERE    dpt = 'CS'
  ```

• SQL returns a bag (multiset) of numbers in which a number may appear several times, provided different faculty had the same salary.

• SQL does not eliminate duplicates, in general, because:
  – Duplicates are important for aggregate queries (e.g., average)
  – Duplicate elimination takes nlogn time.
Relational Algebra Under Bag Semantics

<table>
<thead>
<tr>
<th>Operation</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>$R_1 \cup R_2$</td>
</tr>
<tr>
<td>Intersection</td>
<td>$R_1 \cap R_2$</td>
</tr>
<tr>
<td>Product</td>
<td>$R_1 \times R_2$</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>
Conjunctive Queries Under Bag Semantics

Chaudhuri & Vardi – 1993
Optimization of \textit{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.
- This problem has turned out to be \textit{much more challenging} than originally perceived.
PROBLEMS

Problems worthy of attack prove their worth by hitting back.

in: Grooks by Piet Hein (1905-1996)
**Query Containment Under Set Semantics**

<table>
<thead>
<tr>
<th>Class of Queries</th>
<th>Complexity of Query Containment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunctive Queries</td>
<td>NP-complete</td>
</tr>
<tr>
<td></td>
<td>Chandra &amp; Merlin – 1977</td>
</tr>
<tr>
<td>Unions of Conjunctive Queries</td>
<td>NP-complete</td>
</tr>
<tr>
<td></td>
<td>Sagiv &amp; Yannakakis - 1980</td>
</tr>
<tr>
<td>Conjunctive Queries with ≠, ≤, ≥</td>
<td>Π₂^p-complete</td>
</tr>
<tr>
<td></td>
<td>Klug 1988, van der Meyden -1992</td>
</tr>
<tr>
<td>First-Order (SQL) queries</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Trakhtenbrot - 1949</td>
</tr>
</tbody>
</table>
Bag Semantics vs. Set Semantics

- 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 $a$.
- $Q^{\text{BAG}}(D)$: Result of evaluating $Q$ on (bag) database $D$.
- $Q_1 \subseteq_{\text{BAG}} Q_2$ if for every (bag) database $D$, we have that $Q_1^{\text{BAG}}(D) \subseteq_{\text{BAG}} Q_2^{\text{BAG}}(D)$.

Fact:
- $Q_1 \subseteq_{\text{BAG}} Q_2$ implies $Q_1 \subseteq Q_2$.
- The converse does not always hold.
Bag Semantics vs. Set Semantics

Fact: $Q_1 \subseteq Q_2$ does not imply that $Q_1 \subseteq_{BAG} Q_2$.

Example:
- $Q_1(x) :- P(x), T(x)$
- $Q_2(x) :- P(x)$

- $Q_1 \subseteq Q_2$ (obvious from the definitions)
- $Q_1 \not\subseteq_{BAG} Q_2$
- Consider the (bag) instance $D = \{P(a), T(a), T(a)\}$. Then:
  - $Q_1(D) = \{a,a\}$
  - $Q_2(D) = \{a\}$, so $Q_1(D) \not\subseteq Q_2(D)$. 
Query Containment under Bag Semantics

• Chaudhuri & Vardi - 1993 stated that:
  Under bag semantics, the containment problem for
  conjunctive queries is $\Pi_2^p$-hard.

• Problem:
  – What is the exact complexity of the containment
    problem for conjunctive queries under bag
    semantics?
  – Is this problem decidable?
Query Containment Under Bag Semantics

• 23 years have passed since the containment problem for conjunctive queries under bag semantics was raised.

• Several attacks to solve this problem have failed.

• At least two technically flawed PhD theses on this problem have been produced.

• Chaudhuri and Vardi have withdrawn the claimed $\Pi_2^p$-hardness of this problem; no one has provided a proof.
Query Containment Under Bag Semantics

- The containment problem for conjunctive queries under bag semantics remains open to date.

- However, progress has been made towards the containment problem under bag semantics for the two main extensions of conjunctive queries:
  - Unions of conjunctive queries
  - Conjunctive queries with $\neq$
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 10th Problem

Hilbert’s 10th Problem – 1900
(10th in Hilbert’s list of 23 problems)

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

In effect, Hilbert’s 10th Problem is:
Find an algorithm for the following problem:
Given a polynomial $P(x_1,...,x_n)$ with integer coefficients, does it have an all-integer solution?
Hilbert’s 10\textsuperscript{th} Problem

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

- **Y. Matiyasevich** – 1971
  (building on M. Davis, H. Putnam, and J. Robinson)
  - Hilbert’s 10\textsuperscript{th} Problem is **undecidable**, hence **no** such algorithm exists.
Hilbert’s 10th Problem

• 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 $p_1 \leq p_2$?
    In other words, is it true that $p_1(a_1,\ldots,a_n) \leq p_2(a_1,\ldots,a_n)$, for all positive integers $a_1,\ldots,a_n$?

• Thus, there is no algorithm for deciding questions like:
  – Is $3x_1^4x_2x_3 + 2x_2x_3 \leq x_1^6 + 5x_2x_3$?
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 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**.
Unions of Conjunctive Queries

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 $P_2(w), P_3(w)$. 
## 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></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 (\neq, \leq, \geq)</td>
<td>(\Pi_2^p)-complete</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vdM - 1992</td>
<td></td>
</tr>
<tr>
<td>First-order (SQL) queries</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
<tr>
<td></td>
<td>Trakhtenbrot - 1949</td>
<td></td>
</tr>
</tbody>
</table>
Conjunctive Queries with ≠

Theorem (Jayram, K …, Vee – 2006):
Under bag semantics, the containment problem for conjunctive queries with ≠ 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 (albeit huge) constant.
Conjunctive Queries with ≠

Proof Idea:
Reduction from a 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 $x_1, \ldots, x_{59}$?
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 $\neq$
  – $P_1(x_1,\ldots, x_{59}) \leq (x_1)^5 P_2(x_1,\ldots, x_{59})$
    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 and reduce Hilbert’s 10th Problem to conjunctive query containment over databases of special form (no inequalities are used!)

**Step 2:** Arbitrary databases Use inequalities ≠ in the 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 \( Q_1(D) \subseteq_{BAG} Q_2(D) \).

**Step 3:** Show that we only need a single relation of arity 2.
Additional Comments

• The reduction uses seven different “control” gadgets.

• In Step 2, inequalities $\neq$ are used in both queries.

• Number of inequalities $\neq$ depends on size of special-form DBs, not counting the tuples in the VALUE table.
  – Hence, the number of inequalities depends on the degree of polynomials and the number 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 $\neq$, $\leq$, $\geq$</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>Trakhtenbrot - 1949</td>
<td></td>
</tr>
</tbody>
</table>
Subsequent Developments

- Some progress has been made towards identifying special classes of conjunctive queries for which the containment problem under bag semantics is decidable.
  - Afrati, Damigos, Gergatsoulis – 2010
    - Projection-free conjunctive queries.
  - Kopparty and Rossman – 2011
    - A large class of boolean conjunctive queries on graphs.
The Containment Problem for Boolean Queries

• **Note:**
  For boolean conjunctive queries, the containment problem under bag semantics is equivalent to the Homomorphism Domination Problem.

• **The Homomorphism Domination Problem for graphs**
  Given two graphs G and H, is it true that
  \[
  \# \text{Hom}(G,T) \leq \# \text{Hom}(H,T), \text{ for every graph } T?
  \]
  (where,
  - \# \text{Hom}(G,T) = \text{number of homomorphisms from G to T}
  - \# \text{Hom}(H,T) = \text{number of homomorphisms from H to T}.)
The Homomorphism Domination Problem

Theorem (Kopparty and Rossman – 2011):

• There is an algorithm to decide, given a series-parallel graph \( G \) and a chordal graph \( H \), whether or not \( \# \text{Hom}(G,T) \leq \# \text{Hom}(H,T) \), for all directed graphs \( T \).

Equivalently,

• The conjunctive query containment problem \( Q_1 \subseteq_{\text{BAG}} Q_2 \) is decidable for boolean conjunctive queries \( Q_1 \) and \( Q_2 \) such that the canonical database \( D^{Q_1} \) is a series-parallel graph and the canonical database \( D^{Q_2} \) is a chordal graph.

Note:
The proof using conditional entropy and linear programming.
Question: What is the complexity of conjunctive query evaluation and of conjunctive query equivalence under bag semantics?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Set Semantics</th>
<th>Bag Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQ Evaluation Combined Complexity / Query Complexity</td>
<td>NP-complete</td>
<td>#P-complete</td>
</tr>
<tr>
<td>CQ Equivalence</td>
<td>NP-complete</td>
<td>GRAPH ISOMORPHISM - complete</td>
</tr>
<tr>
<td>CQ Containment</td>
<td>NP-complete</td>
<td>Open</td>
</tr>
</tbody>
</table>
Backup Slides
Conjunctive Queries with ≠

Theorem: Jayram, K …, Vee – 2006
Under bag semantics, the containment problem for conjunctive queries with ≠ 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 (albeit huge) constant.
Conjunctive Queries with ≠

Proof Idea:
Reduction from a 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 $x_1,\ldots,x_{59}$?
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 $\neq$
  - $P_1(x_1,\ldots, x_{59}) \leq (x_1)^5 P_2(x_1,\ldots, x_{59})$
    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 and reduce Hilbert’s $10^{th}$ Problem to conjunctive query containment over databases of special form (**no** inequalities are used!)

**Step 2:** Arbitrary databases
Use inequalities $\neq$ in the 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 $Q_1(D) \subseteq BAG Q_2(D)$.

**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

- $P(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2$
  - $x_1 = 3, x_2 = 2, \ P(3,2) = 3^2 + 3 \cdot 2 + 2^2 = 19.$

- 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:
  - TERM: $\ (X_1,X_1,T_1), \ (X_1,X_2,T_2), \ (X_2,X_2,T_3)$
  - 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:
  - $t \rightarrow T_1$, $u_1 \rightarrow X_1$, $u_2 \rightarrow X_1$.
    Hence:
    $v_1 \rightarrow 1,2,$ or $3$ and $v_2 \rightarrow 1$ or $2$, so we get $3^2$ witnesses.
  - $t \rightarrow T_2$, $u_1 \rightarrow X_1$, $u_2 \rightarrow X_2$.
    Hence:
    $v_1 \rightarrow 1,2,$ or $3$ and $v_2 \rightarrow 1$ or $2$, so we get $3 \cdot 2$ witnesses.
  - $t \rightarrow T_3$, $u_1 \rightarrow X_2$, $u_2 \rightarrow X_2$.
    Hence:
    $v_1 \rightarrow 1$ or $2$, and $v_2 \rightarrow 1$ 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 ≠-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 $\neq$ in the 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 $Q_1(D) \subseteq_{\text{BAG}} Q_2(D)$ 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, \(Q_1(D) \subseteq_{\text{BAG}} Q_2(D)\).
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 \( (X_2,X_1,T') \), then both \( Q_1 \) and \( Q_2 \) can use it equally.
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 $59^{10}$).