The Query Containment Problem: Set Semantics vs. Bag Semantics

Phokion G. Kolaitis
University of California Santa Cruz
&
IBM Research - Almaden
PROBLEMS

Problems worthy of attack prove their worth by hitting back.

in: Grooks by Piet Hein (1905-1996)
An Old Problem in Database Theory

- Database theory research has been going on for more than four decades.
- Over the years, it has had numerous successes.
- Yet, in spite of concerted attacks, some problems have been “hitting back” and resisting solution.

- This talk is about the *conjunctive query containment problem under bag semantics*, an old, but persistent problem that remains open to date.
- This problem was introduced exactly 20 years ago by Surajit Chaudhuri and Moshe Y. Vardi.
- This talk is dedicated to them.
Outline of the Talk

- Background and motivation
- Query containment under set semantics
- Query containment under bag semantics
  - Problem description
  - Partial progress to date
- Concluding remarks and outlook.
Let $Q_1$ and $Q_2$ be two database queries.

- $Q_1 \subseteq Q_2$ means that for every database $D$, we have that $Q_1(D) \subseteq Q_2(D)$, where $Q_i(D)$ is the set of all tuples returned by evaluating $Q_i$ on $D$.

- The Query Containment Problem asks: given two queries $Q_1$ and $Q_2$, is $Q_1 \subseteq Q_2$?

- For boolean queries ("true" or "false), query containment amounts to logical implication $Q_1 \models Q_2$, which is a fundamental problem in logic.
The Query Containment Problem

- Encountered in several different areas, including
  - Query processing
    - query equivalence reduces to query containment:
      \[ Q_1 \equiv Q_2 \text{ if and only if } Q_1 \subseteq Q_2 \text{ and } Q_2 \subseteq Q_1. \]
  - Decision-support
    - \( Q_1 \) may be much easier to evaluate than \( Q_2 \).
    - If \( Q_1 \subseteq Q_2 \), then
      \( Q_1 \) provides a sound approximation to \( Q_2 \).

- Tight connections with constraint satisfaction (but this is another talk).
Complexity of Query Containment

The Query Containment Problem:
Given queries $Q_1, Q_2$, is $Q_1 \subseteq Q_2$?
In other words:
Is $Q_1(D)$ contained in $Q_2(D)$, for all databases $D$?

**Note:** Can’t just try every database $D$ – infinitely many!

**Trakhtenbrot’s Theorem** (1949):
The set of finitely valid first-order sentences is undecidable.

**Corollary:** For first-order queries, the query containment problem is undecidable.
Extensive study of the query containment problem for conjunctive queries and their extensions.

- **Conjunctive queries**: the most frequently asked queries. They are the `SELECT-PROJECT-JOIN` queries.

- **Unions** of conjunctive queries.

- **Conjunctive queries with inequalities** ≠ and arithmetic comparisons ≤ and ≥.
Conjunctive Queries and Their Extensions

- **Conjunctive Query:**
  - $Q(x_1,\ldots,x_k): \exists z_1 \ldots \exists z_m \varphi(x_1,\ldots,x_k,z_1,\ldots,z_m),$
    - where $\varphi$ is a conjunction of atoms.

- **Example:**
  - $TAUGHT-By(x,y): \exists z (ENROLLS(x,z) \land TEACHES(y,z))$
  - Written as a logic rule:
    - $TAUGHT-By(x,y): \Leftarrow ENROLLS(x,z), TEACHES(y,z)$

- **Union of Conjunctive Queries**
  - **Example:** Path of length at most 2:
    - $Q(x,y): E(x,y) \lor \exists z (E(x,z) \land E(z,y))$

- **Conjunctive Query with $\neq$**
  - **Example:** At least two different paths of length 2:
    - $Q(x,y): \exists z \exists w (E(x,z) \land E(z,y) \land E(x,w) \land E(w,y) \land z \neq w).$
Complexity of Conjunctive Query Containment

- **Theorem:** Chandra and Merlin – 1977
  For conjunctive queries, the containment problem is NP-complete.

- **Note:**
  - NP-hardness: reduction from 3-Colorability
  - Membership in NP is not obvious.
    It is a consequence of the following result.
Complexity of Conjunctive Query Containment

**Theorem:** Chandra and Merlin – 1977
For Boolean conjunctive queries $Q_1$ and $Q_2$, the following are equivalent:
- $Q_1 \subseteq Q_2$.
- There is a homomorphism $h : D[Q_2] \rightarrow D[Q_1]$, where $D[Q_i]$ is the canonical database of $Q_i$.

**Example:** Conjunctive query and canonical database
- $Q$: $E(x,y), E(y,z), E(z,x)$
- $D[Q] = \{ E(X,Y), E(Y,Z), E(Z,Y) \}$
Theorem: Sagiv & Yannakakis - 1980
The query containment problem for unions of conjunctive queries is NP-complete.

Note:
- Clearly, this problem is NP-hard, since it is at least as hard as conjunctive query containment.
- Membership in NP is not obvious.
  - It is a consequence of the following result.
Unions of Conjunctive Queries

**Theorem:** Sagiv & Yannakakis - 1980
For all conjunctive queries $Q_1, \ldots, Q_n$, $Q'_1, \ldots, Q'_m$, the following two statements are equivalent:
- $Q_1 \cup \ldots \cup Q_n \subseteq Q'_1 \cup \ldots \cup Q'_m$.
- For every $i \leq n$, there is $j \leq m$, such that $Q_i \subseteq Q'_j$.

**Note:**
- The proof uses the Chandra-Merlin Theorem.
- For membership in NP:
  - we first guess $n$ pairs $(Q'_{k_i}, h_{k_i})$; then
  - we verify that for every $i \leq n$, the function $h_{k_i}$ is a homomorphism from $D[Q'_{k_i}]$ to $D[Q_i]$. 
Conjunctive Queries with Arith. Comparisons

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

  Suffices to test containment on exponentially many “canonical” databases.
- van der Meyden – 1992:
  $\Pi_2^p$-hardness, even for conjunctive queries with only ≠.
The Complexity Class $\Pi_2^p$

- $\Pi_2^p$ is a complexity class that is sandwiched between NP and PSPACE, i.e.,
  \[ \text{NP} \subseteq \Pi_2^p \subseteq \text{PSPACE}. \]

- The prototypical $\Pi_2^p$-complete problem is $\forall \exists \text{SAT}$, i.e., the restriction of QBF to formulas of the form
  \[ \forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n \varphi. \]
## Complexity of Query Containment

<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 $\neq$, $\leq$, $\geq$</td>
<td>$\Pi_2^p$-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>
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*, since they return *bags* (duplicates are *not* eliminated).
A *Real* Conjunctive Query

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

- SQL keeps duplicates, because:
  - Duplicates are important for aggregate queries.
  - 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 $R_1 \cup R_2$</td>
<td>$m_1 + m_2$</td>
</tr>
<tr>
<td>Intersection $R_1 \cap R_2$</td>
<td>$\text{min}(m_1, m_2)$</td>
</tr>
<tr>
<td>Product $R_1 \times R_2$</td>
<td>$m_1 \times m_2$</td>
</tr>
<tr>
<td>Projection and Selection</td>
<td>Duplicates are not eliminated</td>
</tr>
</tbody>
</table>

- $R_1$:
  - $A\ B$
  - 1 2
  - 1 2
  - 2 3

- $R_2$:
  - $B\ C$
  - 2 4
  - 2 5

- $(R_1 \bowtie R_2)$:
  - $A\ B\ C$
  - 1 2 4
  - 1 2 4
  - 1 2 5
  - 1 2 5
Bag Semantics

Chaudhuri & 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 containment under bag 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 \( Q_1 \subseteq_{\text{BAG}} Q_2 \).

Example:

- \( Q_1(x) \) :- \( P(x), \ T(x) \)
- \( Q_2(x) \) :- \( P(x) \)

- \( Q_1 \subseteq Q_2 \) (obvious from the definitions)
- \( Q_1 \not\subseteq_{\text{BAG}} Q_2 \)
- 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**?
20 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 flawed PhD theses on this problem have been produced.

No proof of the claimed $\Pi^p_2$-hardness of this problem has been provided.
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 ≠
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 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?

- Matiyasevich – 1971
  - Hilbert’s 10\textsuperscript{th} Problem is \textit{undecidable}, hence \textbf{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>Undecidable</td>
</tr>
<tr>
<td>Unions of conj. queries</td>
<td>NP-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td>Conj. queries with ≠, ≤, ≥</td>
<td>Π₂^p-complete</td>
<td>Undecidable</td>
</tr>
<tr>
<td>First-order (SQL) queries</td>
<td>Undecidable</td>
<td>Undecidable</td>
</tr>
</tbody>
</table>
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 (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.
Step 1: DBs of a Special Form - Example

- Encode a homogeneous, 2-variable, degree 2 polynomial in which all coefficients are 1.
  \[ P(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 \]
- 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 \(Q \Leftarrow \) TERM\((u_1, u_2, t)\), VALUE\((u_1, v_1)\), 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 \( Q :- \) TERM\((u_1,u_2,t)\), VALUE\((u_1,v_1)\), 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:** \( P(3,2) = 19 = Q^{BAG}(D) \)
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 $\neq$-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 $\neq$ 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

- \( P_1(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 \)
- \( Poly_1(u_1, u_2, t) :- \) TERM\((u_1, u_2, t)\), VALUE\((u_1, v_1)\), VALUE\((u_2, v_2)\)
  the query encoding \( P_1 \) on special-form DBs.
  - 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: \((X_1, 1)\), \((X_1, 2)\), \((X_1, 3)\), \((X_2, 1)\), \((X_2, 2)\), \((T_0, T_0)\)

- \( Q_1 :- Poly_1(u_1, u_2, t) \)
- \( Q_2 :- Poly_2(u_1, u_2, t)\), \( Poly_1(w_1, w_2, w)\), \( w \neq T_1\), \( w \neq T_2\), \( w \neq T_3\)

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 $59^{10}$).
## 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>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 $\# \text{Hom}(G,T) \leq \# \text{Hom}(H,T)$, for every graph $T$? (where,
  - $\# \text{Hom}(G,T) = \text{number of homomorphisms from } G \text{ to } T$
  - $\# \text{Hom}(H,T) = \text{number of homomorphisms from } H \text{ to } T$.}


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:

Sophisticated proof using entropy and linear programming.
Concluding Remarks

- Twenty years after it was first raised and in spite of considerable efforts, the containment problem for conjunctive queries under bag semantics remains open.

- Let us hope that this problem will be settled some time in the next … twenty years.

- But let us also recall another piece of wisdom by Piet Hein.
T.T.T.

Put up in a place
where it is easy to see
the cryptic admonishment
T.T.T.

When you feel how depressingly
slowly you climb
it’s well to remember that
Things Take Time.

in: Grooks by Peter Hein