Constraint Satisfaction and Database Theory

Moshe Y. Vardi*

Rice University
http://www.cs.rice.edu/∼vardi

*A personal perspective

Constraint Satisfaction

WWW index: 10,007 hits on Altavista (vs. 6076 for "database theory")

Applications:

- belief maintenance
- machine vision
- natural language processing
- planning and scheduling
- temporal reasoning
- type reconstruction
- bioinformatics
- ...

Scientific Presence: a conference and a journal.

Constraint Satisfaction Problem (CSP)

Input: (V, D, C):

- A finite set V of *variables*
- A finite set D of *values*
- A finite set C of *constraints* restricting the values that tuples of variables can take.

Constraint: (t, R)

- t: a tuple of variables over V
- R: a relation of arity $|t|$

Solution: $h : V \to D$

- $h(t) \in R$: for all $(t, R) \in C$

Question: Does (V, D, C) have a solution? I.e., is there an assignment of values to the variables such that all constraints are satisfied?

3-Colorability

3-COLOR: Given an undirected graph $A = (V, E)$, is it 3-colorable?

- The variables are the nodes in V.
- The values are the elements in $\{\mathbf{R}, \mathbf{G}, \mathbf{B}\}$.
- The constraints are $\{(\langle u, v\rangle, \rho) \quad : \quad (u, v) \in E\}$, where $\rho = \{(R, G), (R, B), (G, R), (G, B), (B, R), (B, G)\}$.

Database Perspective of CSP

Given: $(V, D, \{C_1, \ldots, C_m\})$, where $C_i = (t_i, R_i)$.

Assume (wlog): Each t_i consists of distinct elements.

Database Perspective:

- V: attributes
- D: values
- (t_i, R_i): relation R_i over relation scheme t_i

Fact: (Bibel, Gyssens, Jeavons, Cohen)

$(V, D, \{C_1, \ldots, C_m\})$ has a solution iff $\bowtie_1^m R_i$ is nonempty.

Homomorphisms

Homomorphism: Let $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ and $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ be two relational structures.

$h : A \to B$ is a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ if for every $i \leq m$ and every tuple $(a_1, \ldots, a_n) \in A^n$,

$$R_i^{\mathbf{A}}(a_1, \ldots, a_n) \implies R^{\mathbf{B}}(h(a_1), \ldots, h(a_n)).$$

The Homomorphism Problem: Given relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism $h : \mathbf{A} \to \mathbf{B}$?

Example: An undirected graph $\mathbf{A} = (V, E)$ is 3-colorable

$$\iff$$

there is a homomorphism $h : \mathbf{A} \to K_3$, where K_3 is the *3-clique*.

Homomorphism Problems

Examples:

- k-Clique: $K_k \overset{h}{\to} (V, E)$?
- Hamiltonian Cycle: $(V, C_{|V|}, \neq) \overset{h}{\to} (V, E, \neq)$?
- Subgraph Isomorphism: $(V, E, \overline{E}) \overset{h}{\to} (V', E', \overline{E'})$?
- s-t Connectivity: $(V, E, \{\langle s, t\rangle\}) \overset{h}{\not\to} (\{0, 1\}, =, \neq)$?

Fact: (Levin, 1973)

The homomorphism problem is NP-complete.

CSP vs. Homomorphisms

From CSP to Homomorphism:

Given: $(V, D, \{C_1, \ldots, C_m\})$, where $C_i = (t_i, R_i)$.

Define $\mathbf{A}, \mathbf{B}$:

- $\mathbf{A} = (V, \{t_1\}, \ldots, \{t_m\})$
- $\mathbf{B} = (D, R_1, \ldots, R_m)$

Fact: (V, D, C) has a solution iff there is homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

CSP vs. Homomorphisms

From Homomorphism to CSP:

Given: $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$, $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

Define (V, D, C):

- $V = A$: elements of $\mathbf{A}$ are variables.
- $D = B$: elements of $\mathbf{B}$ are values.
- $C = \{(t, R_i^{\mathbf{B}}) \ : \ t \in R_i^{\mathbf{A}}\}$: constraints derived from $\mathbf{A}, \mathbf{B}$.

Fact: There is homomorphism from $\mathbf{A}$ to $\mathbf{B}$ iff (V, D, C) has a solution.

Conclusion: CSP=Homomorphism Problem

- Feder&V., 1993
- Garey&Johnson, 1979: Homomorphism in, CSP not.

Conjunctive Queries

Conjunctive Query: First-order logic without $\forall, \vee, \neg$; written as a rule

$$Q(X_1, \ldots, X_n) \ :- \ R(X_3, Y_2, X_4), \ldots, S(X_2, Y_3)$$

Significance: most common SQL queries (*Select-Project-Join*)

Example:

$$GrandParent(X, Y) \ :- \ Parent(X, Z), Parent(Z, Y)$$

Conjunctive Query Containment (CQC)

Definition: $Q_1 \sqsubseteq Q_2$ iff $Q_1(B) \subseteq Q_2(B)$ for every database B.

Fundamental problem in database theory:

- Optimization of SPJ Queries
- Information Integration
- Data Warehousing
- ...

CQC vs. Homomorphism Problem

Canonical Database B^Q:

- Each variable in Q is a distinct element
- Each subgoal $R(X_3, Y_2, X_4)$ of Q gives rise to a tuple $R(X_3, Y_2, X_4)$ in B^Q
- Each *distinguished variable* X_i gives rise to a tuple $P_i(X_i)$ in B^Q

Fact: (Chandra&Merlin, 1977)

For conjunctive queries Q_1 and Q_2, TFAE:

- $Q_1 \sqsubseteq Q_2$
- There is a homomorphism $h : B^{Q_2} \to B^{Q_1}$.
- $Q_2(B^{Q_1})$ is nonempty

Homomorphism Problem vs. CQC

Given: $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$, $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$.

Define conjunctive query $Q_{\mathbf{A}}$ over variables in A:

$$Q_{\mathbf{A}} \ :- \bigwedge_{1}^{m} \bigwedge_{t \in R_i^{\mathbf{A}}} R_i(t)$$

Define conjunctive query $Q_{\mathbf{B}}$ over variables in B:

$$Q_{\mathbf{B}} \ :- \bigwedge_{1}^{m} \bigwedge_{t \in R_i^{\mathbf{B}}} R_i(t)$$

Fact: For relational structures $\mathbf{A}$ and $\mathbf{B}$, TFAE:

- There is a homomorphism $h : \mathbf{A} \to \mathbf{B}$.
- $Q_{\mathbf{B}} \sqsubseteq Q_{\mathbf{A}}$
- $Q_{\mathbf{A}}(\mathbf{B})$ is nonempty

"Same Difference"

Conclusion: (Kolaitis&V., 1998)

CSP, CQ containment, and CQ evaluation amount to the same problem!

Potential Promise: Synergy between AI and DB.

Uniform CSP vs. Non-Uniform CSP

Uniform CSP:

$$\{(\mathbf{A}, \mathbf{B}) : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

Complexity of Uniform CSP: NP-complete

Non-uniform CSP: Fix a structure $\mathbf{B}$

$$\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \exists \text{ homomorphism } h : \mathbf{A} \to \mathbf{B}\}$$

Complexity of Non-Uniform CSP: Depends on $\mathbf{B}$

- $\mathrm{CSP}(K_2)$ is in PTIME (2-Colorability)
- $\mathrm{CSP}(K_3)$ is NP-complete (3-Colorability)

Database Perspective

Uniform CSP:

$$\{(\mathbf{A}, \mathbf{B}) : Q_{\mathbf{A}}(\mathbf{B}) \text{ is nonempty}\}$$

Complexity of Uniform CSP: combined complexity

Non-uniform CSP: Fix a structure $\mathbf{B}$

$$\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : Q_{\mathbf{A}}(\mathbf{B}) \text{ is nonempty}\}$$

Complexity of Non-Uniform CSP: expression complexity

Complexity of CQC

In Conjunctive Query Containment ($Q_1 \sqsubseteq Q_2$?), we are typically interested in *uniform* results.

Research Program:

Identity the tractable cases of uniform CQC.

Results: Tractable cases of $Q_1 \sqsubseteq Q_2$

- Saraiya, 1991:

  Every predicate occurs at most *twice* in Q_1

- Chekuri–Rajaraman, 1997:

  Q_2 has *querywidth* at most k, where k is a fixed positive integer.

- Gottlob–Leone–Scarcello, 1999:

  Q_2 has *hypertreewidth* at most k, where k is a fixed positive integer.

Complexity of Non-Uniform CSP

Research Program:

Identity the tractable cases of non-uniform CSP (Schaefer, 1978, . . .,Jeavons et al)

Dichotomy Conjecture: (Feder–V., 1993)

For every structure $\mathbf{B}$,

- either $\mathrm{CSP}(\mathbf{B})$ is in PTIME
- or $\mathrm{CSP}(\mathbf{B})$ is NP-complete.

Recall: $P \neq NP \Rightarrow NP - NPC - P \neq \emptyset$ (Ladner, 1975)

"Evidence": (Hell–Nešetril, 1990)

Let $\mathbf{B}$ be an *undirected* graph.

- $\mathbf{B}$ bipartite $\Longrightarrow$ $\mathrm{CSP}(\mathbf{B})$ is in PTIME
- $\mathbf{B}$ non-bipartite $\Longrightarrow$ $\mathrm{CSP}(\mathbf{B})$ is NP-complete

Complexity of Non-Uniform CSP

More "Evidence": (Schaefer, 1978)

Let $\mathbf{B}$ have a *Boolean* domain, then

- either $\mathrm{CSP}(\mathbf{B})$ is in PTIME
- or $\mathrm{CSP}(\mathbf{B})$ is NP-complete.

Classification Question:

For a given structure $\mathbf{B}$,

- when is $\mathrm{CSP}(\mathbf{B})$ in PTIME?
- when is $\mathrm{CSP}(\mathbf{B})$ NP-complete?

Classification Conjecture: (Feder–V., 1993)

Two explanations for tractability of $\mathrm{CSP}(\mathbf{B})$

- *combinatorial* (Datalog)
- *algebraic* (group-theoretic)

Datalog and Non-Uniform CSP

Example: Non 2-Colorability

$$\begin{aligned} O(X,Y) &:- E(X,Y) \\ O(X,Y) &:- O(X,Z), E(Z,W), E(W,Y) \\ Q &:- O(X,X) \end{aligned}$$

Recall: $\text{Datalog} \subseteq \text{PTIME}$

Define: $\overline{\text{CSP}(\mathbf{B})} = \{\mathbf{A} : \mathbf{A} \notin \text{CSP}(\mathbf{B})\}$.

Datalog vs. Non-Uniform CSP: Explanation for many tractability results

- $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog

Note: $\overline{\text{CSP}(\mathbf{B})}$ is positively monotone.

k-Datalog

Definition:

- k-Datalog: Datalog with at most k variables per rule (NON 2-COLORABILITY is in 4-Datalog)
- $\exists\mathrm{IL}^k$: k-variable existential positive infinitary logic

Facts: Fix $k \geq 1$

- k-Datalog $\subset$ $\exists\mathrm{IL}^k$
- $\exists\mathrm{IL}^k$ can be characterized in terms of *existential k-pebble games* between the *Spoiler* and the *Duplicator*.
- There is a PTIME algorithm to decide whether the *Spoiler* or the *Duplicator* wins the existential k-pebble game.

Existential k-Pebble Games

$\mathbf{A}, \mathbf{B}$: structures

- **Spoiler**: *places* on or *removes* a pebble from an element of $\mathbf{A}$.
- Duplicator: tries to duplicate move on $\mathbf{B}$.

$\mathbf{A}$: $a_1, a_2, \ldots, a_l$ $\qquad\qquad l \leq k$
$\mathbf{B}$: $b_1, b_2, \ldots, b_l$

- *Spoiler wins*: $h(a_i) = b_i,\ 1 \leq i \leq l$ is not a homomorphism.
- *Duplicator wins*: otherwise.

Fact: (Kolaitis&V., 1995)

$\mathbf{B}$ satisfies the same $\exists \mathrm{IL}^k$ sentences as $\mathbf{A}$ iff the Duplicator wins the existential k-pebble game on $\mathbf{A}, \mathbf{B}$.

k-Datalog and CSP

Theorem: (Kolaitis&V., 1998): TFAE for $k \geq 1$ and a structure $\mathbf{B}$:

- $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog
- $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in $\exists \mathrm{IL}^k$
- $\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \textit{Duplicator}$ wins the existential k-pebble game on $\mathbf{A}$ and $\mathbf{B}\}$.

Intuition: $\overline{\mathrm{CSP}(\mathbf{B})} \in k$-Datalog implies that existence of homomorphism is equivalent to the Duplicator winning the existential k-pebble game.

Winning Configurations

Let $\mathbf{A} = (A, R_1^{\mathbf{A}}, \ldots, R_m^{\mathbf{A}})$ and $\mathbf{B} = (B, R_1^{\mathbf{B}}, \ldots, R_m^{\mathbf{B}})$ be two relational structures. Let $\mathbf{a} \in A^k$ and $\mathbf{a} \in B^k$.

Definition:

- $(\mathbf{a}, \mathbf{b})$ is a *winning configuration* for the Duplicator if the Duplicators wins the existential k-pebble game starting from the position $(\mathbf{a}, \mathbf{b})$.

- $W^k(\mathbf{A}, \mathbf{B})$ is the set of winning configurations.

Theorem: (Kolaitis&V., 1998)

- $W^k(\mathbf{A}, \mathbf{B})$ is definable in least-fixpoint logic over the combined structure $\mathbf{A} + \mathbf{B}$.

- For a fixed structure $\mathbf{B}$, the complement of $W^k(\mathbf{A}, \mathbf{B})$ is definable in k-Datalog over the structure $\mathbf{A}$.

Corollary: For a fixed structure $\mathbf{B}$, there is a k-Datalog program $\rho_{\mathbf{B}}$ such that $\rho_{\mathbf{B}}(\mathbf{A})$ is nonempty iff the Spoiler wins the existential k-pebble game on $\mathbf{A}, \mathbf{B}$.

k-Datalog and CSP

$\rho_{\mathbf{B}}$:

- If $\rho_{\mathbf{B}}(\mathbf{A})$ is nonempty, then $\mathbf{A} \notin \mathrm{CSP}(\mathbf{B})$.
- If $\overline{\mathrm{CSP}(\mathbf{B})}$ is definable in k-Datalog, then it is definable by $\rho_{\mathbf{B}}$.
- *Open question*: Decide for a given $\mathbf{B}$ whether $\overline{\mathrm{CSP}(\mathbf{B})}$ is definable by $\rho_{\mathbf{B}}$.

Database Perspective of k-Datalog: Fix $\mathbf{B}$

$$\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \rho_{\mathbf{B}}(\mathbf{A}) \text{ is nonempty}\}$$

Complexity of Non-Uniform CSP: data complexity

Earlier Database Perspective

Non-uniform CSP: Fix a structure $\mathbf{B}$

$$\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : Q_{\mathbf{A}}(\mathbf{B}) \text{ is nonempty}\}$$

Complexity of Non-Uniform CSP: expression complexity

Winning Strategies

$\mathbf{A}, \mathbf{B}$: structures

Definition: A *winning strategy* for the Duplicator in the existential k-pebble game on $\mathbf{A}, \mathbf{B}$ is a nonempty family $\mathcal{F}$ of k-partial homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ that has the k-*forth property*:

- for every $f \in \mathcal{F}$ with $|f| < k$ and every $a \in A$ on which f is undefined, there is a $g \in \mathcal{F}$ that extends f and is defined on a.

Intuition:

- Think of f as a position in the game.
- If Spoiler places pebble on a, Duplicators responds with a pebble on $g(a)$.

CSP and Consistency

Key Idea in AI Research: *consistency*

- extensibility of partial solutions
- backtrack-free search

Definition: A homomorphism instance $\mathbf{A}, \mathbf{B}$ is

- i-*consistent*: For every consistent assignment α to $i-1$ variables $v_1, \ldots, v_{i-1}$ and a variable v_i, there is a consistent extension of α to v_i.
- *Strongly* k-*consistent*: i-consistency holds for all $i \leq k$.

Game-Theoretic Formulation: (Kolaitis&V., 2000)

$\mathbf{A}, \mathbf{B}$ is strongly k-consistent iff the family of all k-partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$ is a winning strategy for the Duplicator in the existential k-pebble game on $\mathbf{A}, \mathbf{B}$.

From Local to Global Consistency

Condition for Tractability: (Dechter, 1992)

Establishing strong k-consistency implies existence of solution.

- Start with $\mathbf{A}, \mathbf{B}$.
- Generate a strongly k-consistent instance $\mathbf{A}', \mathbf{B}'$ over the same domains.
- If h is a k-partial homomorphism from $\mathbf{A}'$ to $\mathbf{B}'$, then h is a k-partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$.
- A function h is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ iff it is a homomorphism from $\mathbf{A}'$ to $\mathbf{B}'$.

Intuition: $\mathbf{A}', \mathbf{B}'$ is obtained by inferring from $\mathbf{A}, \mathbf{B}$ all constraints on k-variables – *constraint propagation*.

Consistency and k-Datalog

Strong k-Consistency vs. Games: (K.&V., 2000)

The strongly k-consistent instance $\mathbf{A}', \mathbf{B}'$ can be obtained by "re-formatting" $W^k(\mathbf{A}, \mathbf{B})$

Intuition: Computing winning strategies can be viewed as constraint propagation.

Corollary: Let $\mathbf{B}$ a relational structure. $\overline{\mathrm{CSP}(\mathbf{B})}$ is expressible in k-Datalog iff for every structure $\mathbf{A}$, establishing strong k-consistency for $\mathbf{A}, \mathbf{B}$ implies that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

Constraint Propogation

A database perspective on constraint propagation:

- *Projection Rule*: From (t, R) infer $(t, \pi_{t'}(R))$, where $t' \subseteq t$.

- *Join Rule*: From (t_1, R_1) and (t_1, R_2), infer $(t_1 \cup t_2, R_1 \bowtie R_2)$

Eaxmple: *Resolution* – From $\alpha = \theta \vee p$ and $\beta = \psi \vee \neg p$ infer $\theta \vee \psi$

- Sound and complete for refutation of clausal formulas

Soundness Proof:

- Consider $models(\varphi)$ as a relation (over a Boolean domain) with relation scheme $vars(\varphi)$.

- $\pi_{vars(\theta) \cup vars(\psi) - \{p\}} (models(\alpha) \bowtie models(\beta)) = models(\theta \vee \psi)$

Bounded Treewidth

Definition: A *tree decomposition* of a structure $\mathbf{A} = (A, R_1, \ldots, R_m)$ is a labeled tree T such that

- Each label is a non-empty subset of A;
- For every R_i and every $(a_1, \ldots, a_n) \in R_i$, there is a node whose label contains $\{a_1, \ldots, a_n\}$.
- For every $a \in A$, the nodes whose label contain a form a subtree.

The *treewidth* $\mathrm{tw}(\mathbf{A})$ of $\mathbf{A}$ is defined by

$$\mathrm{tw}(\mathbf{A}) \;=\; \min_{T}\{\max\{\text{label size in } T\}\} - 1$$

Note: Generalizes the *treewidth* of a *graph*.

Bounded Treewidth and CSP

$$C_k = \{\mathbf{A} : \ \mathrm{tw}(\mathbf{A}) \leq k\}$$

Theorem: (Freuder, 1990)

$\mathrm{CSP}(C_k, \mathrm{All})$ is in PTIME.

Note:

- Complexity is exponential in k.
- Determining treewidth of $\mathbf{B}$ is NP-hard.
- Checking if treewidth is k is in linear time.

Database Perspective of Bounded Treewidth

Tractable cases: $\mathrm{CSP}(\mathbf{A}, \mathbf{B})$

- Yannakakis, 1981:

  $\mathbf{A}$ is *acyclic*

- Freuder, 1990:

  $\mathbf{A}$ has *treewidth* at most k

- Chekuri–Rajaraman, 1997:

  $\mathbf{A}$ has *querywidth* at most k

- Gottlob–Leone–Scarcello, 1999:

  $\mathbf{A}$ has *hypertreewidth* at most k

Acyclicity is the crux: (Flum–Frick–Grohe, 2000)

The above can all be viewed as instances of Yannakakis's algorithm.

Complexity of Query Evaluation

Expression Complexity: Fix $\mathbf{B}$

$$\{Q : Q(\mathbf{B}) \text{ is nonempty}\}$$

Data Complexity: Fix Q

$$\{\mathbf{B} : Q(\mathbf{B}) \text{ is nonempty}\}$$

Exponential Gap: (V., 1982)

- Data complexity of FO: *LOGSPACE*
- Expression complexity of FO: *PSPACE-complete*

Mystery: practical query evaluation

Variable-Confined Queries

Definition: FO^k is first-order logic with at most k variables.

In Practice: (V., 1995)

- Queries often can be rewritten to use a small number of variables.
- Variable-confined queries have lower expression complexity.
- E.g.: expression complexity of FO^k is *PTIME-complete*

Recall: $\mathbf{A} \mapsto Q_{\mathbf{A}}$

- $h : \mathbf{A} \to \mathbf{B}$ iff $Q_{\mathbf{A}}(\mathbf{B})$ is nonempty

Theorem: (Kolaitis&V., 1998)

$\mathbf{A}$ has treewidth k iff $Q_{\mathbf{A}}$ is expressible in existential, positive FO with $k+1$ variables.

AI vs. Databases

Fundamentally different approaches:

- *AI*: search one answer
- *Databases*: generate all answers

Fundamental Question:: Is query evaluation effective for solving CSP?

Recall: $\mathbf{A} \mapsto Q_{\mathbf{A}}$

- $h : \mathbf{A} \to \mathbf{B}$ iff $Q_{\mathbf{A}}(\mathbf{B})$ is nonempty

Experiment: (with Alfonso San Miguel Aguirre)

- Express 3-SAT as a CSP.
- Translate to conjunctive query evaluation.
- Run on DB2.

Outcome: Terrible performance!!!

- DB2 optimizer ill suited.

View-Based Query Processing

Basic Idea: Computing the answer to a query based on a set of views, rather than on the raw data in the database.

Significance: Relevant problem in query optimization, query answering with incomplete information, data warehousing, data integration, etc.

Two approaches:

- View-based query rewriting
- View-based query answering

View-Based Query Tewriting

Basic Idea: Given a query Q and a set of view definitions, reformulate Q into a new query expressed in a given language on the alphabet of the view names.

Many previous results:

- conjunctive queries
- queries with aggregates
- recursive queries
- Description Logics queries
- queries for semi-structured data

But: no general definition of query rewriting!!!

View-Based Query Answering

Basic Idea: Given a query Q, a set of view definitions, and a set of associated view extensions, compute the set of tuples t that are in the answer to Q in all the databases consistent with the views (*certain* tuples).

Many previous results:

- complexity for several query and view languages [Abiteboul et al 98], [Grahne et al 99]
- Description Logics queries [Calvanese et al, AAAI 2000]
- Regular path queries [Calvanese et al, ICDE 2000]

Query Answring vs. Query Rewriting

Query Answering: $\mathrm{cert}(Q, \mathrm{def}(\mathcal{V}), \mathrm{ext}(\mathcal{V})$ computes the set of certain tuples for the query Q wrt to the view definition $\mathrm{def}(\mathcal{V})$ and the view extension $\mathrm{ext}(\mathcal{V})$.

Query Rewriting: (CDLV, 2000)

- Define $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}$ to be the Currying of cert wrt Q and $\mathrm{def}(\mathcal{V})$.
- $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}(\mathrm{ext}(\mathcal{V})$ is the set of certain tuples for the query Q wrt to the view definition $\mathrm{def}(\mathcal{V})$ and the view extension $\mathrm{ext}(\mathcal{V})$.
- $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}$ *is* the rewriting of Q wrt $\mathrm{def}(\mathcal{V})$.

Of interest: properties of $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}$

- *Traditional*: expressibility in certain query language
- *New*: computational complexity

Complexity of Query Answering

$\mathrm{cert}(Q, \mathrm{def}(\mathcal{V}), \mathrm{ext}(\mathcal{V})$

Complexity: (CDLV, 2000)

- *Combined complexity*: in terms of Q, $\mathrm{def}(\mathcal{V})$, and $\mathrm{ext}(\mathcal{V})$ – *PSPACE-complete*.
- *Expression complexity*: in terms of Q, and $\mathrm{def}(\mathcal{V})$ – *PSPACE-complete*.
- *Data complexity*: in terms of $\mathrm{ext}(\mathcal{V})$ – *co-NP-complete*.

Note: Complexity of $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}$ is data complexity of cert – *co-NP-complete*.

Questions: When is $\mathrm{cert}_{Q,\mathrm{def}(\mathcal{V})}$ in PTIME? in Datalog?

Query Rewriting and CSP

Intimate Connection: between CSP and query rewriting.

Theorem: (CDLV,2000)

- Non-uniform CSP over directed graphs is polynomially reducible to the evaluation of query rewritings.

- Evaluation of query rewritings is polynomially reducible to non-uniform CSP.

Consequences:

- Classifying tractable query rewritings is closely related to classifying non-uniform CSP – *both hard*.

- Datalog programs for non-uniform CSP can be used as *approximate* query rewritings.

In Conclusion

CSP is a database problem!!!

There is a real potential for AI-DB cross fertilization.