Aspects of Database Query Evaluation

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

Database Theory and Mihalis

- Over the years, Mihalis Yannakakis has made a number of highly influential and long-lasting contributions to the principles of database systems.

- The aim of this talk is to present an overview of some of these contributions (and of subsequent developments) with emphasis on Mihalis' work on database query evaluation.

The Relational Data Model

E.F. Codd, 1969-1971

- Relational Schema

  Sequence **S** = $(R'_1,\ldots,R'_m)$ of relation symbols of specified arities.

- Relational Database over **S** :

  Collection D = $(R_1, \ldots, R_m)$ of finite relations (tables) of matching arities.

- Database Query Languages:
  - Relational Calculus (First-Order Logic)
  - Relational Algebra.

Database Queries

- Informally, database queries are questions that are posed against a database, and answers are retrieved.

- A k-ary query, $k \geq 1$, on a relational schema **S** is a function Q such that on every database D over **S**, Q(D) is a k-ary relation.

  Examples:

  ENROLLS(student,course), TEACHES(faculty, course)

  - TAUGHT-BY = { (s,f): s is enrolled in some course taught by f }
  - FAN-OF = { (s,f): s is enrolled in every course taught by f }

- Boolean query: a 0-ary query; it returns value 1 or 0.

  Examples:

  - Is there a student enrolled in four different courses?
  - Is there a faculty who teaches only one course?

Database Query Languages

- A query language is a formalism for expressing queries.

- Codd introduced two different query languages, a declarative one and a procedural one.
  - Relational Calculus: A query is given by a formula of first-order logic with quantifiers ranging over elements occurring in relations in the database.
  - Relational Algebra: A query is given by an expression involving the operations projection π, selection σ, cartesian product $\times$, union $\cup$, and set-difference $\setminus$.

- Codd showed that these two query languages have the same expressive power.

- SQL: The standard commercial database query language is based on relational algebra and relational calculus.

Expressing Database Queries

ENROLLS(student,course), TEACHES(instructor, course)

- TAUGHT-BY(student,instructor)
  - Relational calculus expression (first-order formula)
    $\exists c\ (\mathrm{ENROLLS}(s,c) \wedge \mathrm{TEACHES}(f,c))$
  - Relational algebra expression
    $\pi_{1,3}(\sigma_{\$2=\$4}(\mathrm{ENROLLS} \times \mathrm{TEACHES}))$

- FAN-OF(student,instructor)
  - Relational calculus expression
    $\forall c\ (\mathrm{TEACHES}(f,c) \rightarrow \mathrm{ENROLLS}(s,c))$

The Query Evaluation Problem

The Query Evaluation Problem:
Given a query Q and a database D, compute Q(D).

- k-ary query, $k \geq 1$: Q(D) is the k-ary relation consisting of all tuples of values from D that satisfy the query.
- Boolean query: Q(D) is 1 or 0
  - Q(D) = 1 if D satisfies Q (denoted by $D \models Q$)
  - Q(D) = 0 if D does not satisfy Q.

Note: The Query Evaluation Problem is arguably the most fundamental problem in database query processing.

Complexity of Query Evaluation

Fact: The query evaluation problem for relational calculus/relational algebra is PSPACE-complete.

Reason:

- Upper bound: Alternating polynomial-time algorithm
- Lower bound: Reduction from QBF.

Question: Are there “useful” fragments of relational calculus/relational algebra for which the query evaluation problem is of lower complexity?

Enter Conjunctive Queries

Conjunctive Queries:

- Are among the most frequently asked database queries.
- Are expressible by syntactically very simple formulas of first-order logic.
- Are the SELECT-PROJECT-JOIN queries of relational algebra.
- Are directly supported in SQL.

Conjunctive Queries

Conjunctive Query of arity $k \geq 1$:

$Q(x_1,\ldots,x_k)$: $\exists z_1 \ldots \exists z_m \, \varphi(x_1,\ldots,x_k,z_1,..z_m)$,

where φ is a conjunction of atoms $R_i(y_1,\ldots,y_m)$

- Example: TAUGHT-BY

TAUGHT-BY(s,f): $\exists$ c(ENROLLS(s,c) $\wedge$ TEACHES(f,c))

- Example: Path of length 3:

P3(x,y): $\exists z \, \exists w \, (E(x,z) \wedge E(z,w) \wedge E(w,y))$

Boolean Conjunctive Query

Q( ): $\exists x_1 \ldots \exists x_n \, \varphi(x_1,\ldots,x_n)$

- Example: Is there a triangle?

C3( ): $\exists x \, \exists y \, \exists z \, (E(x,y) \wedge E(y,z) \wedge E(z,x))$

Conjunctive Queries and SQL

Fact: SQL provides direct support for conjunctive queries

Example: Consider the conjunctive query

- TAUGHT-BY(s,f): $\exists$ c(ENROLLS(s,c) $\wedge$ TEACHES(f,c))

  Recall that

  TAUGHT-BY = $\pi_{1,3}(\sigma_{\$2=\$4}(\text{ENROLLS} \times \text{TEACHES}))$

- SQL expression for this query:

  SELECT student, instructor

  FROM ENROLLS, TEACHES

  WHERE ENROLLS.course = TEACHES.course

(**SELECT = π; WHERE = σ; FROM = $\times$**)

More on Conjunctive Queries

Recall also the query

FAN-OF(student,instructor),

which is expressible by the first-order logic formula

$\forall c\,(\text{TEACHES}(f,c) \rightarrow \text{ENROLLS}(s,c))$

Fact: FAN-OF is not equivalent to any conjunctive query

Reason:

- Conjunctive queries are monotone.
- FAN-OF is not monotone.

The Conjunctive Query Evaluation Problem

The Conjunctive Query Evaluation Problem:

Given a conjunctive query Q and a database D, compute Q(D).

Theorem: Chandra and Merlin – 1977

The conjunctive query evaluation problem is NP-complete.

Proof:

- NP-hardness: Reduction from CLIQUE
  - G contains a k-clique iff $G \models \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} E(x_i, x_j)$
- Membership in NP is a consequence of the following result.

Complexity of Conjunctive Query Evaluation

Theorem: Chandra and Merlin – 1977
Boolean Conjunctive Query Evaluation is "equivalent" to the Homomorphism Problem. More precisely,

For a Boolean conjunctive query Q and a database D, the following statements are equivalent:

- $D \models Q$ (i.e., Q(D) = 1).
- There is a homomorphism $h : D^Q \rightarrow D$, where D^Q is the canonical database of Q.

Example: Conjunctive query and canonical database

- Q( ): $\exists x \exists y \exists z(E(x,y) \wedge E(y,z) \wedge E(z,x))$
- $D^Q = \{ E(X,Y), E(Y,Z), E(Z,Y) \}$

Islands of Tractability

Major Research Program:
Identify tractable cases of conjunctive query evaluation.

Note:
Over the years, this program has been pursued by two different research communities:

- The Database Theory community.
- The Constraint Satisfaction community.
  Explanation:
  As pointed out by Feder & Vardi (1993),
  the Constraint Satisfaction Problem can be identified
  with the Homomorphism Problem.

A Large and Useful Island of Tractability

- In 1981, Mihalis Yannakakis discovered a large and useful tractable case of the Conjunctive Query Evaluation Problem.

Specifically,

- Mihalis showed that the Query Evaluation Problem is tractable for Acyclic Conjunctive Queries.

Acyclic Conjunctive Queries

Definition: A conjunctive query Q is acyclic if it has a join tree.

Definition: Let Q be a conjunctive query of the form

$$Q(\mathbf{x}) : \exists \mathbf{y} \, (R_1(\mathbf{z}_1) \wedge R_2(\mathbf{z}_2) \wedge \ldots \wedge R_m(\mathbf{z}_m)).$$

A join tree for Q is a tree T such that

- The nodes of T are the atoms $R_i(\mathbf{z}_i)$, $1 \leq i \leq m$, of Q.
- For every variable w occurring in Q, the set of the nodes of T that contain w forms a subtree of T;

  in other words, if a variable w occurs in two different atoms $R_j(\mathbf{z}_j)$ and $R_k(\mathbf{z}_k)$ of Q, then it occurs in each atom on the unique path of T joining $R_j(\mathbf{z}_j)$ and $R_k(\mathbf{z}_k)$.

Acyclic Conjunctive Queries

- Path of length 4 is acyclic

  $P4(x_1,x_4) : \exists x_2\ x_3\ (E(x_1,x_2) \wedge E(x_2,x_3) \wedge E(x_3,x_4))$

  - Join tree is a path

- Cycle of length 4 is cyclic

$C4(\,) : \exists x_1\ x_2\ x_3\ x_4 (E(x_1,x_2) \wedge E(x_2,x_3) \wedge E(x_3,x_4) \wedge E(x_4,x_1))$

- The following query Q is acyclic

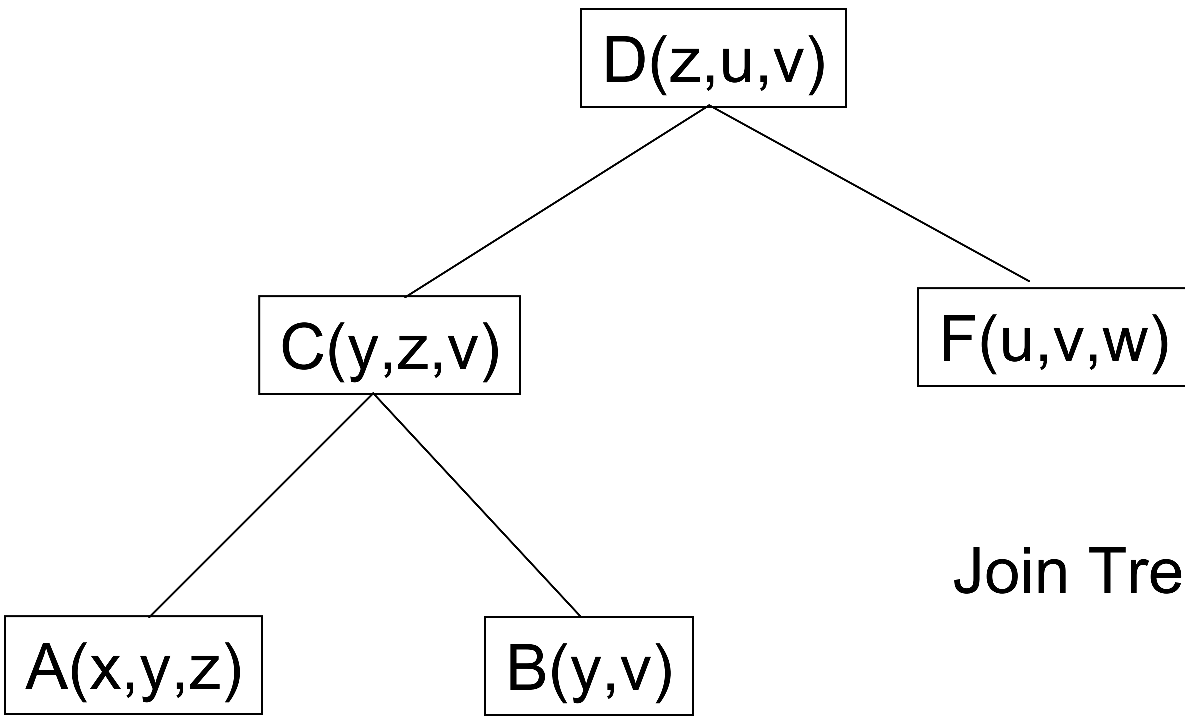
$Q(\,) : \exists x\ y\ z\ u\ v\ w$

$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$

Acyclic Conjunctive Queries

Q( ) : $\exists$ x y z u v w

$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$

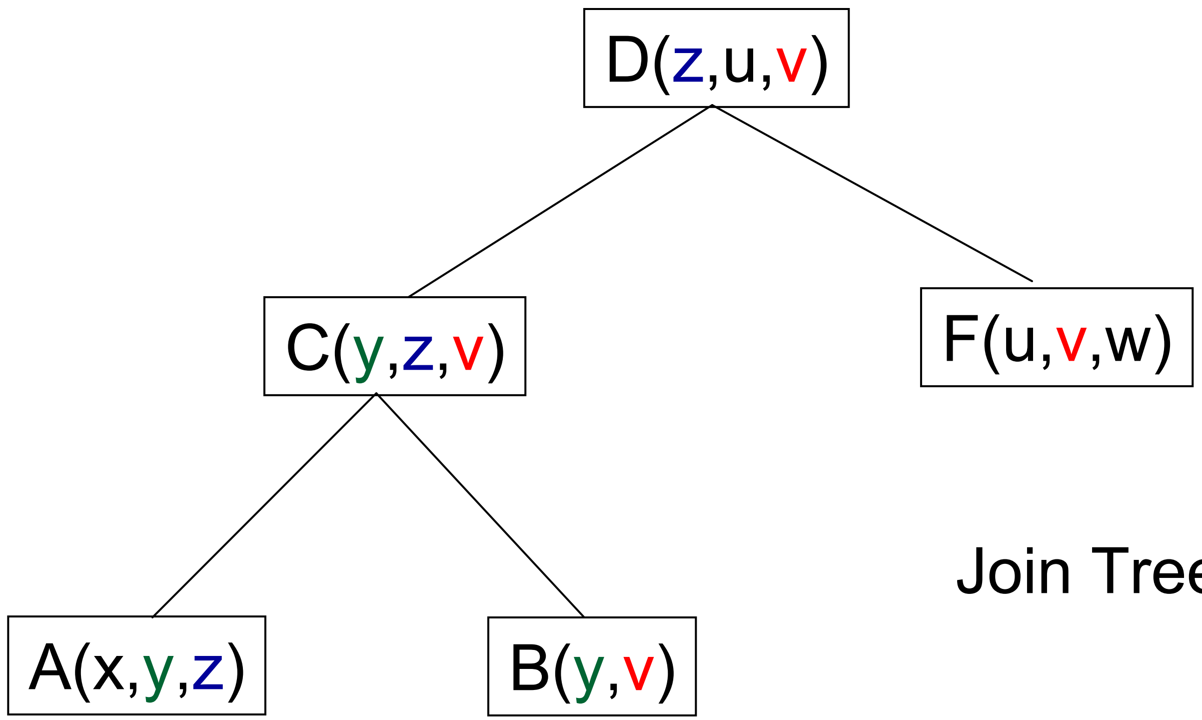


Join Tree for Q

Acyclic Conjunctive Queries

$Q(\,): \exists\, x\, y\, z\, u\, v\, w$

$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$



Join Tree for Q

Acyclic Conjunctive Queries

Theorem (Yannakakis – 1981)
The Acyclic Conjunctive Query Evaluation Problem is tractable.
More precisely, there is an algorithm for this problem having the following properties:

- If Q is a Boolean acyclic conjunctive query, then the algorithm runs in time O(|Q||D|).

- If Q is a k-ary acyclic conjunctive query, $k \geq 1$, then the algorithm runs in time O(|Q||D||Q(D)|), i.e., it runs in input/output polynomial time (which is the “right” notion of tractability in this case).

Yannakakis' Algorithm

Dynamic Programming Algorithm

Input: Boolean acyclic conjunctive query Q, database D

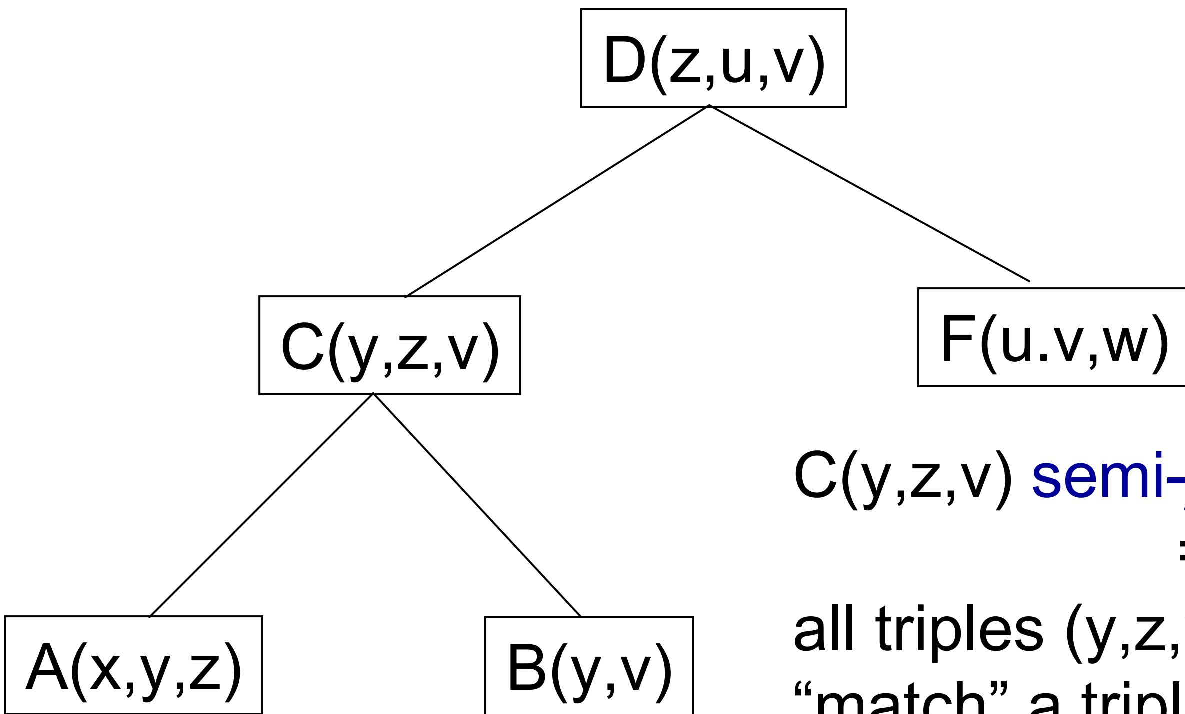
1. Construct a join tree T of Q
2. Populate the nodes of T with the matching relations of D.
3. Traverse the tree T bottom up:

   For each node $R_k(z_k)$, compute the semi-joins of the (current) relation in the node $R_k(z_k)$ with the (current) relations in the children of the node $R_k(z_k)$.
4. Examine the resulting relation R at the root of T
   - If R is non-empty, then output Q(D) = 1 (D satisfies Q).
   - If R is empty, then output Q(D) = 0 (D does not satisfy Q).

Yannakakis' Algorithm

Q( ) : $\exists$ x y z u v w

$(A(x,y,z) \wedge B(y,v) \wedge C(y,z,v) \wedge D(z,u,v) \wedge F(u,v,w))$



C(y,z,v) semi-join A(x,y,z)

=

all triples (y,z,v) in C that "match" a triple (x,y,z) in A

More on Yannakakis' Algorithm

- The join tree makes it possible to avoid exponential explosion in intermediate computations.

- The algorithm can be extended to non-Boolean conjunctive queries using two more traversals of the join tree.

- There are efficient algorithms for detecting acyclicity and computing a join tree.
  - Tarjan and Yannakakis – 1984

  Linear-time algorithm for detecting acyclicity and computing a join tree.

  - Gottlob, Leone, Scarcello – 1998

  Detecting acyclicity is in SL (hence, it is in L).

Subsequent Developments

Yannakakis' algorithm became the catalyst for numerous subsequent investigations in different directions, including:

- Direction 1: Identify the exact complexity of Boolean Acyclic Conjunctive Query Evaluation.
  - Yannakakis' algorithm is sequential (e.g., if the join tree is a path of length n, then n-1 semi-joins in sequence are needed).
  - Is Boolean Acyclic Conjunctive Query Evaluation P-complete? Is it in some parallel complexity class?

- Direction 2: Identify other tractable cases of Conjunctive Query Evaluation.

Complexity of Acyclic Conjunctive Query Evaluation

Theorem (Dalhaus – 1990)
Boolean Acyclic Conjunctive Query Evaluation is in NC^2 .

Theorem (Gottlob, Leone, Scarcello - 1998)
Boolean Acyclic Conjunctive Query Evaluation is LOGCFL-complete, where LOGCFL is the class of all problems having a logspace-reduction to some context-free language.

Fact:

- $NL \subseteq LOGCFL \subseteq AC^1 \subseteq NC^2 \subseteq P$
- LOGCFL is closed under complements (Borodin et al. - 1989)

Tractable Conjunctive Query Evaluation

- Extensive pursuit of tractable cases of conjunctive query evaluation during the past three decades.

- Two different branches of investigation
  - The relational vocabulary **S** is fixed in advance;

    in this case, the input conjunctive query is over **S**.
  - Both the relational schema and the query are part of the input.

- Note that in Yannakakis' algorithm both the relational schema and the query are part of the input.

Enter Tree Decompositions and Treewidth

Definition: **S** a fixed relational schema, D a database over **S**.

- A tree decomposition of D is a tree T such that
  - Every node of T is labeled by a set of values from D.
  - For every relation R of D and every tuple $(d_1,\ldots d_m) \in R$, there is a node of T whose label contains $\{d_1, \ldots, d_m\}$.
  - For every value d in D, the set X of nodes of T whose labels include d forms a subtree of T.

- width(T) = max(cardinality of a label of T) – 1

- Treewidth: tw(D) = min {width(T): T tree decomposition of D}

Conjunctive Queries and Treewidth

Definition: **S** a fixed relational schema,
Q a Boolean conjunctive query over **S**.

- tw(Q) = tw(Q^D), where
  Q^D is the canonical database of Q.

- **TW**(k,**S**) = All Boolean conjunctive queries Q over **S** with
  tw(Q) $\leq$ k.

Note: Fix a relational schema **S**.

- If Q is a Boolean acyclic conjunctive query over **S**, then
  tw(Q) $\leq$ max {arity(R): R is a relation symbol of **S**} - 1.

- The converse is not true. For every n $\geq$ 3, the query
  Cn = "is there a cycle of length n?" is cyclic, yet tw(Cn) = 2.

Conjunctive Queries and Treewidth

Theorem (Dechter & Pearl – 1989, Freuder 1990)

- For every relational schema **S** and every k $\geq$ 1, the query evaluation problem for **TW**(k,**S**) is tractable.
- In words, there is a polynomial-time algorithm for the following problem: given a database D and a Boolean conjunctive query Q over **S** of treewidth at most k, does D $\models$ Q?

Note:

This result was obtained in the quest for islands of tractability of the Constraint Satisfaction Problem.

Beyond Bounded Treewidth

Question: Are there islands of tractability for conjunctive query evaluation larger than bounded treewidth?

Definition: Two queries Q and Q are equivalent, denoted $Q \equiv Q'$, if Q(D) = Q'(D), for every database D.

Fact: Let Q and Q be Boolean conjunctive queries. Then $Q \equiv Q'$ if and only if D^Q and $D^{Q'}$ are homomorphically equivalent, i.e., there are homomorphisms h: $D^Q \rightarrow D^{Q'}$ and h': $D^{Q'} \rightarrow D^Q$.

Note: This follows from the Chandra-Merlin Theorem.

Beyond Bounded Treewidth

Definition: **S** a fixed relational schema,
Q a Boolean conjunctive query over **S**.

- **HTW**(k,**S**) = All Boolean conjunctive queries Q over **S** such that Q $\equiv$ Q', for some Q' $\in$ **TW**(k,**S**).

Fact: Q $\in$ **HTW**(k,**S**) if and only if core(Q) $\in$ **TW**(k,**S**),
where core(Q) is the minimization of Q, i.e.,
the smallest subquery of Q that is equivalent to Q.

Note: **TW**(k,**S**) is properly contained in **HTW**(k,**S**)
Reason:
The k $\times$ k grid has treewidth k, but it is 2-colorable, hence it is homomorphically equivalent to K_2, which has treewidth 1.

Beyond Bounded Treewidth

Theorem (Dalmau, K …, Vardi – 2002)

- For every relational schema **S** and every k $\geq$ 1, the evaluation problem for **HTW**(k,**S**) is tractable.
- In words, there is a polynomial-time algorithm for the following problem: given a database D and a Boolean conjunctive query Q that is equivalent to some conjunctive query of treewidth at most k, does D $\models$ Q?
- In fact, this problem is in Least Fixpoint Logic.

Algorithm:

- Determine the winner in a certain pebble game, known as the existential k-pebble game.
- No tree decomposition is used (actually, computing tree decompositions is hard).

A Logical Characterization of Treewidth

Definition: **S** a relational vocabulary, k positive integer.
L^k is the collection of all first-order formulas with k variables, containing all atoms of **S**, and closed under $\wedge$ and $\exists$.

Theorem (Dalmau, K …, Vardi – 2002)
S a relational schema, Q a Boolean conjunctive query over **S**.
Then the following are equivalent:

- Q $\in$ **HTW**(k,**S**)
- core(Q) $\in$ **TW**(k,**S**)
- Q is equivalent to some L^{k+1}-sentence.

Example: The query Cn : "is there a cycle of length n?" can be expressed in L^3. For instance, C5 is equivalent to
$\exists x(\exists y(E(x,y) \wedge \exists z\, (E(y,z) \wedge \exists y\, (E(z,y) \wedge \exists z\, (E(y,z) \wedge E(z,x))))))$

The Largest Islands of Tractability

Question: Are there islands of tractability larger than **HTW**(k,**S**)?

Answer: "No", modulo a complexity-theoretic hypothesis.

Theorem (Grohe – 2007)
Assume that FPT $\neq$ W[1].
Let **S** be a relational vocabulary and **C** a recursively enumerable collection of Boolean conjunctive queries over **S** such that the query evaluation problem for **C** is tractable. Then there is a positive integer k such that **C** $\subseteq$ **HTW**(k,**S**).

Proof: Use the Excluded Grid Theorem by Robertson & Seymour

Fixed vs. Variable Relational Schemas

- The preceding results assume that we have a fixed relational schema **S**, and the conjunctive queries are over **S**.

- As mentioned earlier, in Yannakakis' algorithm both the relational schema and the query are part of the input.

- When the relational schema is part of the input, then acyclic queries may have (cores of) unbounded treewidth.
  - $Q_n(\,)$: $\exists x_1 \ldots \exists x_n R_n(x_1,\ldots,x_n)$

- Thus, the preceding results do not subsume Yannakakis' work in the case in which the relational schema is part of the input.

Variable Relational Schemas

- Extensive pursuit of tractable cases of conjunctive query evaluation when the relational schema is part of the input.
  - Several extensions of treewidth have been explored.
  - Hypertree decomposition notions have been studied.

- Chekuri & Rajaraman – 1997: query-width

- Gottlob, Leone, Scarcello – 2000 on: hypertree-width:
  - Acyclicity amounts to hypertree-width = 1.
  - Tractable conjunctive query evaluation for conjunctive queries of bounded hypertree-width.

- No analog of Grohe's Theorem for this set-up has been found.

Combined Complexity vs. Data Complexity

- In the definition of the query evaluation problem, the input consists of the query and the database.

- In 1982, Vardi introduced a useful taxonomy in the study of the query evaluation problem.
  - Combined Complexity of Query Evaluation:

    The input consists of the query and the database.
  - Data Complexity of Query Evaluation:

    A separate problem for each fixed query Q:

    Given a database Q, compute Q(D).

Combined Complexity vs. Data Complexity

Fact: The combined complexity of Boolean conjunctive query evaluation is NP-complete (restating Chandra & Merlin – 1997).

Fact: The data complexity of Boolean conjuctive query evaluation is in AC_0. In other words:
For each fixed Boolean conjunctive query Q, the following problem is in AC_0: given a database D, does $D \models Q$?

Note:

- The low data complexity of conjunctive query evaluation is often viewed as an explanation as to why database systems can efficiently evaluate conjunctive queries.
- However, this is not the end of the story of query evaluation.

Parameterized Complexity

Theorem (Papadimitriou & Yannakakis – 1997)
For both fixed and variable relational schemas,
and with the query size as the parameter:

- The parameterized complexity of conjunctive query evaluation is W[1]-complete.
- The parameterized complexity of relational calculus query evaluation is W[t]-hard, for all t.

Note: Several subsequent investigations of the parameterized complexity of query evaluation by

- Downey, Fellows and Taylor
- Flum, Frick and Grohe
- …

Database Theory and Mihalis

- Mihalis' work in database theory extends well beyond the query evaluation problem. In fact, over the years, he has contributed to a number of different areas, including
  - Database transactions
  - Concurrency control
  - Database design
  - Datalog.

- Database theory is a meeting point of algorithms, complexity, graph theory, and logic. Mihalis' contributions to database theory have been long lasting and influential.