A Call to Regularity

Moshe Y. Vardi*

Rice University

*Joint work with D. Calvanese, G. De Giacomo, and M. Lenzerini


Metamucil
#1 DOCTOR RECOMMENDED WITH
100% NATURAL PSYLLIUM FIBER
FIBER LAXATIVE/DIETARY FIBER SUPPLEMENT
SMOOTH TEXTURE
ARTIFICIALLY FLAVORED
ORANGE FLAVOR
Sugar Free
See back for nutrition information on sugar
and calorie content. Not a low calorie product.
Fill controlled by weight, not volume
NET WT 15 OZ
(425 g)
72 TEASPOON DOSES

Database Query Languages

- Standard database query languages (e.g., SQL 2.0) are essentially 1st-order.

- Aho and Ullman, 1979: 1st-order languages are weak; add *recursion*

- Gallaire and Minker, 1978: add recursion via *logic programs*

- SQL 3.0, 1999: recursion added

Expressiveness costs money!!!

- 1st-order queries: *LOGSPACE*

- Recursive queries: *PTIME*

Datalog

Datalog:

- Function-free logic programs
- Existential, positive fixpoint logic
- Select-project-join-union-recur queries

Example: *Transitive Closure*

$$Path(x, y) \; :- \; Edge(x, y)$$

$$Path(x, y) \; :- \; Path(x, z), Path(z, y)$$

Definition: A program P is *bounded* if it is equivalent to a non-recursive program.

Example: Impressionable Shopper

$$Buys(x, y) \; :- \; Trendy(x), Buys(z, y)$$

$$Buys(x, y) \; :- \; Likes(x, y)$$

Data Complexity

Definitions:

- The *stage function* $s_P(n)$ of a program P is the least m such that $P^m(D) = P^\infty(D)$ for each D with at most n elements.

- A query Q is in $STAGE(f(n))$ if it is expressible by a program P such that $s_P(n)$ is in $O(f(n))$.

Database complexity and computational complexity:

- $STAGE(\text{polylog } n) \subseteq NC$

- $STAGE(\text{poly } n) \subseteq PTIME$

Gap Theorem [Kanellakis, 1992]:

- *P is bounded iff it defines a query in $STAGE(1)$*

- *P is unbounded iff s_P is in $\Omega(\log n)$*

Gaifman, Mairson, Sagiv, V., 1987: Boundedness is undecidable.

Research Program - Study Boundary

Parameters:

- Number of derived predicates
- Arity of derived predicates
- Number of rules
- Nonlinear vs. linear (one recursive call per rule)
- I/O convention

GMSV: undecidability holds for linear programs with a single 4-ary derived predicate.

Binary Programs

Binary programs: binary derived predicates.

Theorem [Hillebrand, Kanellakis, Mairson, V., 1995]: *Boundedness is undecidable for programs with a single binary derived predicate.*

Proof: Redcution from halting problem for Turing machines:

- Σ: tape alphabet
- *Base predicates*: $Zero(x), Succ(x, y), Q_a(x)$ for $a \in \Sigma$
- *Derived predicates*: $Fing(x, y)$– pointers to corresponding positions in successive configurations

Cosmadakis, Gaifman, Kanellakis, V., 1988: Boundedness is decidable for unary programs.

Uniform Boundedness

I/O Conventions:

- **Directed I/O**: *input* – base predicates, *output* – derived predicate.
- **uniform I/O**: *input/output* – all predicates: a program slice

Uniform boudedness: boundedness with respect to the uniform I/O convention.

Sirup: Single recursive rule program

Undecidability of uniform boundness:

- [GMSV, 1987]: 7-ary programs
- [HKMV, 1995]: 3-ary programs
- [Abiteboul, 1989]: sirups
- [Marcinkowski, 1996]: 3-ary sirups, 3-ary linear programs

Query Containment

Query Optimization: Given Q, find Q' such that:

- $Q \equiv Q'$
- Q' is "easier" than Q

Query Containment: $Q_1 \sqsubseteq Q_2$ if $Q_1(B) \subseteq Q_2(B)$ for all databases B.

Fact: $Q \equiv Q'$ iff $Q \sqsubseteq Q'$ and $Q' \sqsubseteq Q$

Consequence: Query containment is a *key* database problem.

Query Containment

Other applications:

- query reuse
- query reformulation
- information integration
- cooperative query answering
- integrity checking
- ...

Consequence: Query containment is a *fundamental* database problem.

Decidability of Query Containment

- *SQL*: undecidable
  - Folk Theorem
  - Poor theory and practice of optimization
- SPJU: decidable
  - Chandra&Merlin–1977, Sagiv&Yannakakis–1982
  - Rich theory and practice of optimization
- Datalog: undecidable
  - Shmueli–1977
  - Difficult theory and practice of optimization

Unfortunately, most decision problems involving Datalog are undecidable - almost no interesting, well-behaved fragments.

1990s: Back to Binary Relations

WWW:

- Nodes
- Edges
- Labels

Semistructured Data: WWW, SGML documents, library catalogs, XML documents, Meta data, ...

Formally: (D, E, λ)

- D - nodes
- $E \subseteq D^2$ - edge
- $\lambda : E \to \Lambda^+$ – labels (alt., also node labels)

Path Queries

Active Research Topic: What is the right query language for semistructured data?

Basic Element of all proposals: *path queries*

- $Q(x, y) \ :- \ x \ L \ y$
- L: formal language over labels
- $a \cdot \underline{\ \ l_1\ \ } \cdots \underline{\ \ l_k\ \ } \cdot b$
- $Q(a, b)$ holds if $l_1 \cdots l_k \in L$

Example: *Regular Path Query*

$Q(x, y) \ :- \ x \ (Wing \cdot Part^+ \cdot Nut) \ y$

Path-Query Containment

$Q_1(x, y) \ :- \ x \ L_1 \ y$

$Q_2(x, y) \ :- \ x \ L_2 \ y$

Language-Theoretic Lemma 1:

$$Q_1 \sqsubseteq Q_2 \text{ iff } L_1 \subseteq L_2$$

Proof: Consider a database

$a \cdot \xrightarrow{l_1} \cdots \xrightarrow{l_k} \cdot b$ with $l_1 \cdots l_k \in L_1$

Corollary: Path-Query Containment is

- undecidable for context-free path queries
- decidable for regular path queries.

Regular Path Queries

Observations:

- A fragment of Transitive-Closure Logic
- A fragment of binary Datalog
  - Concatenation: $E(x,y) \;:-\; E_1(x,z), E_2(z,y)$
  - Union: $E(x,y) \;:-\; E_1(x,y)$
    $E(x,y) \;:-\; E_2(x,y)$
  - Transitive Closure: $P(x,y) \;:-\; E(x,z)$
    $P(x,y) \;:-\; E(x,z), E(z,y)$

Consequence:

- Data complexity: *NLOGSPACE*
- Expression complexity: *PTIME*

Containment: PSPACE-complete, via nondeterministic automata (Stockmeyer, 1973).

Language Containment – Upper Bound

Lemma: $L(E_1) \subseteq L(E_2)$ iff $L(E_1) - L(E_2)) = \emptyset$

Algorithm for checking whether $L(E_1) \subseteq L(E_2)$:

1. Construct NFAs A_i such that $L(A_i) = L(E_i)$ – *linear blow-up*.

2. Construct $\overline{A_2}$ such that $L(\overline{A_2}) = \Sigma^* - L(A_2)$ – *exponential blow-up*.

3. Construct $A = A_1 \times \overline{A_2}$ such that $L(A) = L(E_1) - L(E_2)$ – *quadratic blow-up*.

4. Check if there is a path from start state to final state in A – *NLOGSPACE*.

Bottom Line: *PSPACE*

Two-Way RPQs

Extended Alphabet: $\Lambda^- = \{a^- \;:\; a \in \Lambda^+\}$

$\Lambda = \Lambda^+ \cup \Lambda^-$

Inverse Roles:

$Part(x, y)$: y part of x

$Part^-(x, y)$: x part of y

Example: Step Siblings

$Q(x, y) \;:-$
$x \;\; [(father^- \cdot father) + (mother^- \cdot mother)]^+ \;\; y$

Containment: Two-way nondeterministic automata

- Hopcroft and Ullman, 1979: 2DFA
- Hopcroft, Motwani and Ullman, 2000: ???

2NFA

$A = (\Sigma, S, S_0, \rho, F)$

- Σ – finite alphabet
- S – finite state set
- $S_0 \subseteq S$ – initial states
- $F \subseteq S$ – final states
- $\rho : S \times \Sigma \rightarrow 2^{S \times \{-1,0,+1\}}$ – transition function

Theorem: Rabin&Scott, Shepherdson, 1959

2NFA $\equiv$ 1NFA

2RPQ Containment

Difficulties:

- 2NFA $\rightarrow$ 1NFA: *exponential blow-up*
  - **Consequence**: Doubly exponential complementation
- Difference between query and language containment
  - $Q_1(x,y) \ :- \ x \ Parent \ y$
    $Q_2(x,y) \ :- \ x \ Parent \cdot Parent^- \cdot Parent \ y$
  - $Q_1 \sqsubseteq Q_2$ but
    $L(Parent) \not\subseteq L(Parent \cdot Parent^- \cdot Parent)$

Back to Basics: 2NFA→1NFA

Theorem: Vardi, 1988

Let $A = (\Sigma, S, S_0, \rho, F)$ be a 2NFA. There is a 1NFA A^c such that

- $L(A^c) = \Sigma^* - L(A)$
- $||A^c|| \in 2^{O(||A||)}$

Proof: Guess a subset-sequence counterexample

$a_0 \cdots a_{k-1} \not\in L(A)$ iff there is a sequence $T_0, T_1, \cdots, T_k$ of subsets of S such that

1. $S_0 \subseteq T_0$ and $T_k \cap F = \emptyset$.
2. If $s \in T_i$ and $(t, +1) \in \rho(s, a_i)$, then $t \in T_{i+1}$, for $0 \leq i < k$.
3. If $s \in T_i$ and $(t, 0) \in \rho(s, a_i)$, then $t \in T_i$, for $0 \leq i < k$.
4. If $s \in T_i$ and $(t, -1) \in \rho(s, a_i)$, then $t \in T_{i-1}$, for $0 < i \leq k$.

Foldings

Definition: Let $u, v \in \Lambda^*$. We say that v *folds* onto u, denoted $v \rightsquigarrow u$, if v can be "folded" on u, e.g.,

$$abb^- bc \rightsquigarrow abc.$$

Pictorially, $\xrightarrow{a} \cdot \xrightarrow{b} \cdot \xleftarrow{b} \cdot \xrightarrow{b} \cdot \xrightarrow{c} \rightsquigarrow \xrightarrow{a} \cdot \xrightarrow{b} \cdot \xrightarrow{c}$

Definition: Let E be an RE over Λ. Then $fold(E) = \{v \ : \ v \rightsquigarrow u, u \in L(E)\}$.

Language-Theoretic Lemma 2:

Let $Q_1(x, y) \ :- \ x \ E_1 \ y$
$Q_2(x, y) \ :- \ x \ E_2 \ y$

be 2RPQs. Then $Q_1 \sqsubseteq Q_2$ iff $L(E_1) \subseteq fold(E_2)$.

2RPQ containment

Theorem: Let E be an RE over Λ. There is a 2NFA $\tilde{A}_E$ such that

- $L(\tilde{A}_E) = fold(E)$
- $||\tilde{A}_E|| \in O(||E||)$

Containment $Q_1(x,y) \ :- \ x \ E_1 \ y$
$Q_2(x,y) \ :- \ x \ E_2 \ y$

TFAE

- $Q_1 \sqsubseteq Q_2$
- $L(E_1) \subseteq fold(E_2)$.
- $L(E_1) \subseteq L(\tilde{A}_{E_2})$.
- $L(E_1) \cap L(\tilde{A}^c_{E_2}) = \emptyset$
- $L(A_{E_1} \times \tilde{A}^c_{E_2}) = \emptyset$

Bottom-line: 2RPQ containment is PSPACE-complete.

View-Based Query Processing

- *Global database*: B over Λ^+
- *Views*: $\{V_1, \ldots, V_n\}$, V_i is a query
- *View extensions*: $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$, $\mathcal{E}_i \subseteq V_i(B)$
- *Global query* Q over Λ
- *Local query* over $V_1, \ldots, V_n$

Query Processing

1. *View-based query answering*: approximate $Q(B)$ using view-extension information.
2. *View-based query rewriting*: approximate global query by a local query based on view definitions
3. View-based query losslessness: Compare global query with its view-based approximation.
4. *View-based query containment*: Compare view-based approximations of two global queries.

View-Based Query Rewriting

- *Global database*: B over Λ^+
- *Views*: $\{V_1, \ldots, V_n\}$, V_i is a query
- *View extensions*: $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$, $\mathcal{E}_i \subseteq V_i(B)$
- *Global query* Q over Λ
- *Local query* over $V_1, \ldots, V_n$

Query Rewriting

$\Delta^+ = \{v_1, \ldots, v_n\}$

$\Delta = \Delta^+ \cup \Delta^-$

- Find regular expression $\mathcal{E}$ over Δ such that $\mathcal{E}[v_i \mapsto V_i, v_i^- \mapsto rev(V_i)] \sqsubseteq Q$.
  - $rev(v) = v^-$, $rev(v^- = v)$, $rev(e_1 + e_2) = rev(e_1) + rev(e_2)$, $rev(e_1; e_2) = rev(e_2); rev(e_1)$, $rev(e^*) = rev(e)^*$
- Find maximal such $\mathcal{E}$.

Example: $Q = abcd$, $V_1 = ab$, $V_2 = cd$: $Q = V_1 V_2$

Counterexample Method

Candidate Rewriting: $w = a_1 \ldots a_k \in \Delta^k$

- w is a *bad* rewriting if $w[v_i \mapsto V_i, v_i^- \mapsto rev(V_i)] \not\sqsubseteq Q$.

- w is a *bad* rewriting if there are *witnesses* $w_1, \ldots, w_k \in \Lambda^*$ such that $w_1 \ldots w_k \not\sqsubseteq L(Q)$, where
  - $w_i \in L(V_j)$ if $a_i = v_j$.
  - $w_i \in L(rev(V_j))$ if $a_i = v_j^-$.

- $a_1 w_1 \ldots a_k w_k$: *counterexample word*

Example: $Q = abcd$, $V_1 = ab$, $V_2 = cd$

- $v_1 v_1$: bad rewriting, $v_1 v_2$: good rewriting

- $w_1 = ab$, $w_2 = ab$: witnesses

- $v_1 w_1 v_1 w_2$: counterexample word

Regular Counterexamples

Counterexample Word: $a_1 w_1 \ldots a_k w_k$

1. $w_i \in L(V_j)$ if $a_i = v_j$.

2. $w_i \in L(rev(V_j))$ if $a_i = v_j^-$.

3. $w_1 \ldots w_k \not\sqsubseteq L(Q)$

Checking counterexample words with 2NFA:

- Check (1) and (2) with 2NFA for V_j

- Use folding technique to construct 2NFA to check $w_1 \ldots w_k \sqsubseteq L(Q)$ and then complement.

Complexity: exponential

From Counterexamples to Rewritings

Constructing Good Rewritings

1. Construct 1NFA A_1 for counterexample words (*exponential*).

2. Project out witness words to get 1NFA A_2 for bad rewritings ($a_1 w_1 \ldots a_k w_k \mapsto a_1 \ldots a_k$) (*linear*).

3. Complement A_2 to get 1NFA A_3 for good rewritings (*exponential*).

Theorem:

- Construction yields maximal rewriting (represented by a 1DFA).

- Doubly expoential complexity is optimal.

- Checking whether the rewriting is equivalent to Q is 2EXPSPACE-complete.

Conjunctive Queries

Conjunctive Query: Existential, conjunctive, positive first-order logic, i.e., first-order logic without $\forall, \vee, \neg$; written as a rule

$$Q(x_1, \ldots, x_n) \ :- \ R_1(x_3, y_2, x_4), \ldots, R_k(x_2, y_3)$$

Significance:

- Most common SQL queries (*Select-Project-Join*)
- Core of Datalog

Example:

$$Triangle(x, y, z) \ :- \ Edge(x, y), Edge(y, z), Edge(z, x)$$

Conjunctive Query Containment

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

Fact: (Chandra and Merlin, 1977)

For conjunctive queries Q_1 and Q_2, TFAE:

- The containment $Q_1 \sqsubseteq Q_2$ holds
- There is a homomorphism $h : B^{Q_2} \to B^{Q_1}$ that is the identity on distinguished variables.

Conjunctive 2RPQ

C2RPQ: Core of all semistructured query languages

$Q(x_1, \ldots, x_n) \;:-\; y_1 E_1 z_1, \ldots, y_m E_m z_m$

- E_i – 2RPQ

Intuition:

- C2RPQs are obtained from CQ by replacing atoms with REs over Λ.
- C2RPQs are Select-Project-"Regular Join" queries.

Example:

$$\begin{aligned} Q(x,y) \;:-\; & z \;\; (Wing \cdot Part^+ \cdot Nut) \;\; x, \\ & z \;\; (Wing \cdot Part^+ \cdot Nut) \;\; y \end{aligned}$$

C2RPQ Containment

Difficulty: Earlier techniques do not apply

- No canonical database
- No language-theoretic lemma

Solution: Combine and extend earlier ideas

- Infinite family of canonical databases
  - Each variable in Q is a distinct element
  - Each subgoal $y_i E_i z_i$ of Q is replaced by a simple path labeled by a word in $L(E_i)$.
- Represent canonical databases as words over a larger alphabet
- Develop automata-theoretic characterization of C2RPQ containment.

Bottom-line: C2RPQ containment is EXPSPACE-complete.

In Conclusion

Regular queries:

- A rich but well-behaved fragment of Datalog
- Of special interest for semistructured data
- Beautiful application of classical formal-language theory
- Novel theory of regular paths in labeled graphs

Research Question: What is the ultimate class of regular queries?

- *RPQs*
- *2RPQs*
- *C2RPQs*
- *UC2RPQs*
- ...