A Theory of Regular Queries

Moshe Y. Vardi

Rice University
Codd’s Two Fundamental Ideas:

- **Tables are relations**: a row in a table is just a tuple; order of rows/tuples does not matter!

- **Formulas are queries**: they specified the what rather then the how – declarative programming!

Codd, 1970: A first-order formula $\varphi(x_1, \ldots, x_k)$ defines a query:

$$\varphi(B) = \{ \langle \alpha(x_1), \ldots, \alpha(x_k) \rangle : G \models_\alpha \varphi(x_1, \ldots, x_k) \}$$

**Example**: $(\exists y)(\exists z)(\neg(y = z) \land E(x, y) \land E(x, z))$ – “List all graph nodes that have at least two distinct neighbors”

Codd, 1971: FOL is “relationally complete”. 
Evolution of Database Query Languages

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

• Zloof, 1976: add transitive closure (QBE).

• Gallaire&Minker, 1978: add recursion via logic programs

• Aho&Ullman, 1979: 1st-order languages are weak – add recursion

• SQL 3.0, 1999: recursion added (“common table expressions”)

Expressiveness/complexity trade-off:

• 1st-order queries: Data complexity – LOGSPACE

• Recursive queries: Data complexity – PTIME
Datalog [Chandra&Harel, 1985, Maier&Warren, 1988]:

- Function-free logic programs
- Existential, positive fixpoint logic

Example: Transitive Closure

\[
\text{Path}(x, y) : \neg \text{Edge}(x, y) \\
\text{Path}(x, y) : \neg \text{Path}(x, z), \text{Path}(z, y)
\]
Query Containment, I

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, II

Other applications:

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

Consequence: Query containment is the fundamental database-reasoning problem.
Decidability of Query Containment:

- **SQL**: undecidable
  - Folk Theorem (unsolvability of FO)
  - Poor theory and practice of optimization

- **UCQ (aka, SPJU Queries)**: decidable
  - Chandra & Merlin, 1977 – CQ
  - Sagiv & Yannakakis, 1982 – UCQ
  - Rich theory and practice of optimization

**UCQ**:

- *Existential positive FO*: conjunction, disjunction, existential quantification
- Nonrecursive Datalog
- Covers a very large fraction of real-life database queries

**Example**: CQ – (conjunction + ∃ quantification)

\[ \text{Triangle}(x, y) : \neg \text{Edge}(x, y), \text{Edge}(y, z), \text{Edge}(z, x) \]
Query Containment, IV

Datalog Containment:

- **Complexity**: undecidable
  - Shmueli–1987 - easy reduction from CFG containment

- **Difficult theory and practice of optimization**

**Observation**: most decision problems involving Datalog are undecidable.

**Reminder**: Datalog = UCQ + Recursion

**Question**: Can we limit recursion to recover decidability?
Monadic Datalog

**Monadic Datalog (MDL):** Datalog where recursive predicates are *monadic* [Cosmadakis et al., 1988]

**Example:**
\[
\begin{align*}
\text{Retrieve}(X) & : - \text{Paper}(X), \text{Author}(X, \text{Moshe}) \\
\text{Coauthor}(A) & : - \text{Retrieve}(X), \text{Author}(X, A) \\
\text{Retrieve}(Y) & : - \text{Paper}(Y), \text{Author}(Y, A), \text{Coauthor}(A)
\end{align*}
\]

**Major Application:** web data extraction [Gottlob et al., 2002–]

**Theorem:** Query Containment for MDL is $2\text{EXPTIME-Complete}$ [Cosmadakis et al., 1988, Benedikt et al., 2012]
Network Datalog

**Declarative Networking**: declarative specification of network protocols – an instance of *software-defined networking*

**Network Datalog (NDlog)**: a variant of Datalog adapted to declarative networking [Loo at al., 2006]

\[
\text{path}(\@S, D, P, C) :- \text{link}(\@S, D, C), P = f\_init(S, D).
\]

\[
\text{path}(\@S, D, P, C) :- \text{link}(\@S, Z, C_1), \text{path}(\@Z, D, P_2, C_2), C = C_1 + C_2, P = f\_concat(S, P_2), f\_inPath(P_2, S) = false.
\]

**Key Feature**: reasoning about transitive reachability

**Corollary**: MDL is too weak for declarative networking.

**Major Open Question**: Find an expressive fragment of Datalog with a decidable query-containment problem.
1990s: Graph Databases

WWW:

- Nodes
- Edges
- Labels

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

Graph Databases: $(D, E, \lambda)$

- $D$ - nodes
- $E \subseteq D^2$ - edges
- $\lambda: E \to \Lambda$ – labels (alt., also node labels)
Figure 1: Graph Database
Path Queries

**Active Research Topic:** What is the right query language for graph databases? (“NoSQL”)

**Basic element of all proposals:** path queries

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

**Example:** *Regular Path Query (RPQ)*

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

- A fragment of Dyadic 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**: \( P(x, y) : - E(x, y) \)
    \[ P(x, y) : - P(x, z), P(z, 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

\[ a \cdot \underbrace{l_1 \cdots l_k}_b \ \text{with} \ l_1 \cdots l_k \in L_1 \]

**Corollary:** Path-Query Containment is essentially language containment.

**Corollary:** Path-Query Containment is *undecidable* for context-free path queries

**RPQ Containment:** What is known about containment of regular languages?
Theory of Regular Languages, I

Regular Languages - Robust Definability:

- Regular expressions
- DFA
- NFA
- 2NFA
- AFA
- 2AFA
- Regular grammar
- MSO
- ...

But: Succinctness Gaps: E.g., NFA<RE, NFA<DFA, AFA<NFA, MSO<AFA, ...
NFA

\[ A = (\Sigma, S, S_0, \rho, F) \]

- **Alphabet:** \( \Sigma \)
- **States:** \( S \)
- **Initial states:** \( S_0 \subseteq S \)
- **Nondeterministic transition function:**
  \[ \rho : S \times \Sigma \rightarrow 2^S \]
- **Accepting states:** \( F \subseteq S \)

**Input word:** \( a_0, a_1, \ldots, a_{n-1} \)

**Run:** \( s_0, s_1, \ldots, s_n \)

- \( s_0 \in S_0 \)
- \( s_{i+1} \in \rho(s_i, a_i) \) for \( i \geq 0 \)

**Acceptance:** \( s_n \in F \)

**Recognition:** \( L(A) \) – words accepted by \( A \).

**Example:**

\[ \begin{array}{c}
1 \\
0 \\
\hline
0 \\
1
\end{array} \]

– ends with 1’s
Theory of Regular Languages, II

Regular Languages - Robust Closure:

- Union
- Intersection
- Complement
- Concatenation
- Kleene star
- Reverse
- Homomorphism
- Inverse homomorphism

...
NFA Intersection

Given:

• $A^1 = (\Sigma, S_1^1, S_0^1, \rho^1, F_1^1)$

• $A^2 = (\Sigma, S_2^2, S_0^2, \rho^2, F_2^2)$

Define: $A^1 \cap A^2 = (\Sigma, S_1^1 \times S_2^2, S_0^1 \times S_0^2, \rho, F_1^1 \times F_2^2)$, where:

• $\rho((s, t), a) =$

  $$\{(s', t') : s \in \rho^1(s, a) \text{ and } t' \in \rho^2(t, a)\}$$
NFA Complementation

Run Forest of automaton $A$ on word $w$:

- Roots: elements of $S_0$.
- Children of $s$ at level $i$: elements of $\rho(s, a_i)$.
- Rejection: no leaf is accepting.

Key Observation: collapse forest into a DAG – at most one copy of a state at a level; width of DAG is at most $|S|$.

Subset Construction Rabin-Scott, 1959:

- $A^c = (\Sigma, 2^S, \{S_0\}, \rho^c, F^c)$
- $F^c = \{T : T \cap F = \emptyset\}$
- $\rho^c(T, a) = \bigcup_{t \in T} \rho(t, a)$
- $L(A^c) = \Sigma^* - L(A)$
Complementation Blow-Up

\[ A = (\Sigma, S, S_0, \rho, F), \; |S| = n \]
\[ A^c = (\Sigma, 2^S, \{S_0\}, \rho^c, F^c) \]

**Blow-Up:** \(2^n\) upper bound

*Can we do better?*

**Lower Bound:** \(2^n\)
Sakoda-Sipser 1978, Birget 1993

\[ L_n = (0 + 1)^*1(0 + 1)^{n-1}0(0 + 1)^* \]
- \(L_n\) is easy for NFA
- \(\overline{L_n}\) is hard for NFA
Regular Languages - Robust Decidability:

Emptiness: \( L(A) = \emptyset \)

Nonemptiness Problem: Decide if given \( A \) is nonempty.

NFA Nonemptiness:

**Directed Graph** \( G_A = (S, E) \) of NFA \( A = (\Sigma, S, S_0, \rho, F) \):
- **Nodes**: \( S \)
- **Edges**:
  \[ E = \{(s, t) : t \in \rho(s, a) \text{ for some } a \in \Sigma\} \]

**Lemma**: \( A \) is nonempty iff there is a path in \( G_A \) from \( S_0 \) to \( F \).

- Decidable in time linear in size of \( A \), using breadth-first search or depth-first search.
- **Complexity**: NLOGSPACE-complete.
NFA Containment

**Containment:** \( L(A_1) \subseteq L(A_2) \)

**Lemma:** \( L(A_1) \subseteq L(A_2) \) iff \( A_1 \cap A_2^c \) is empty.

- Decidable in exponential time.

- **Complexity:** PSPACE-complete [Stockmeyer&Meyer, 1973]

- Result holds also for RE containment.
  - Linear translation from RE to NFA
Path-Query Containment

\[ Q_1(x, y) : \neg x \cdot L_1 y \]
\[ Q_2(x, y) : \neg x \cdot L_2 y \]

**Language-Theoretic Lemma 1:**

\[ Q_1 \subseteq Q_2 \iff L_1 \subseteq L_2 \]

**Corollary:** Path-Query Containment is

- PSPACE-complete for regular path queries, via RE containment.

**Comment:** Current NFA containment tools can handle, in practice, automata with *thousands* of states [Bonchi&Pous, 2015].
Two-Way RPQs

Extended Alphabet: \( \Lambda^- = \{ a^- : a \in \Lambda \} \)
\( \Lambda^\pm = \Lambda \cup \Lambda^- \)

Inverse Roles:

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

Example: \( (1/2)^* \) Siblings

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

Containment: Use 2NFA?

- Hopcroft&Ullman, 1979: 2DFA
- Hopcroft&Motwani&Ullman, 2000: ???
2NFA

\[ A = (\Sigma, S, S_0, \rho, F) \]

- \( \Sigma \) – 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 → 1NFA**: exponential blow-up
  
  – **Consequence**: Doubly exponential complementation

• Difference between query and language containment
  
  – \( Q_1(x, y) : \neg x \text{ Parent } y \)
    \( Q_2(x, y) : \neg x \text{ Parent} \cdot \text{Parent}^- \cdot \text{Parent } y \)

  – \( Q_1 \sqsubseteq Q_2 \) but
    \( L(\text{Parent}) \not\subseteq L(\text{Parent} \cdot \text{Parent}^- \cdot \text{Parent}) \)
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

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

**Corollary:** 2NFA containment is PSPACE-complete.

**But:** Recall that the Language-Theoretic Lemma fails for 2RPQ!
Foldings

**Definition:** Let $u, v \in \Lambda^{\pm*}$. We say that $u$ *folds* onto $v$, denoted $u \rightsquigarrow v$, if $u$ can be “folded” onto $v$, e.g.,

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

Pictorially, \[
\begin{array}{ccccccccc}
 & a & \rightarrow & . & b & \rightarrow & . & b & \leftarrow & . & b & \rightarrow & . & c & \rightsquigarrow & a & \rightarrow & . & b & \rightarrow & . & c
\end{array}
\]

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

**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 $\Lambda^\pm$. There is a 2NFA $\tilde{A}_E$ such that

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

Containment

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

TFAE

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

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


**Closing 2RPQs under \( \cap \) and \( \cup \)**

**Intersection:**

- Regular languages are closed under intersection and union.
- Intersection adds succinctness: \( \text{RE}(\cap) < \text{RE} \)

**Intersection vs. Conjunction:**

\[ Q_1(x, y) : - (xE_1 \cap E_2 y) \]
\[ Q_2(x, y) : - (xE_1 y) \land (xE_2 y) \]

**Conclusion:** Intersection \( \neq \) Conjunction for graph databases!

**UC2RPQ:** Closure of 2RPQs under disjunction (union) and conjunction

**Example:**

\[ Q(x, y) : -(xEy) \]

\[ Q(x, y) : -(xE_1 z) \land (zE_2^* y) \land (xE_3 y) \]
**UC2RPQ**

**UC2RPQ:** Core of all graph query languages

\[ Q(x_1, \ldots, x_n) : - y_1E_1z_1, \ldots, y_mE_mz_m \]

- \( E_i \) – 2RPQ
- No recursion (other than 2RPQs)

**Intuition:**

- UC2RPQs are obtained from UCQ by replacing atoms with REs over \( \Lambda^{\pm} \).
- UC2RPQs are Select-Project-Union-“Regular Join” queries.

**Example:**

\[ Q(x, y) : - z \ (Wing \cdot Part^+ \cdot Nut) \ x, \]
\[ z \ (Wing \cdot Part^+ \cdot Nut) \ y \]
**UC2RPQ Containment**

**Difficulty:** Earlier techniques do not apply

- Database techniques cannot handle transitive closure.
- No language-theoretic lemma to reduce to automata (even with folding).

**Solution:** combine database-theoretic and automata-theoretic techniques:

- Search for a counterexample database, e.g., $Q_1(B) \not\subseteq Q_2(B)$
- Represent database $B$ as a word $w$ over a richer alphabet.
- Use 2NFA to evaluate $Q_1$ and $Q_2$ over $w$.

**Bottom-line:** UC2RPQ containment is EXPSPACE-complete. [Calvanese et al., 2000]
Regular Queries

**UC2RPQs:**

- **Elements:** disjunction, conjunction, and transitive closure
- **Closed Under:** disjunction, conjunction
- **Not closed Under:** transitive closure!

**Example:** Not in UC2RPQ!

\[
Q(x, y) : -(xE_1z) \& (zE_2y) \& (xE_3y)
\]

\[
Answer(x, y) : -(xQ^*y)
\]

**RQ:** closure under disjunction, conjunction, and transitive closure (TC).

- **Essentially:** Non-recursive Datalog + TC

**RQ Containment**

- Decidable - *Nonelementary* (via MSO) [Jugé+V., 2009]

- 2EXPSPACE-complete [Reutter&Romero&V., 2015]
Back to Datalog

\[ RQ \rightarrow \text{Datalog:} \]

- Every construct of RQ can be expressed in Datalog.
- Recursion is used only to express transitive closure.
  - If \( Q(x, y) \) is a binary predicate, then use the rules
    \[
    Q^+(x, y) : \neg Q(x, y)
    \]
    \[
    Q^+(x, z) : \neg Q^+(x, y), Q^+(y, z)
    \]

**In fact:** Over graph databases, RQ is precisely *Dyadic Datalog*, where (1) rule heads are dyadic, and (2) recursion is used only to express transitive closure.
Generalized Regular Queries

2EXPSPACE Upper Bound:

- Automata-theoretic techniques developed over almost 30 years. (“Automata Theory for Database Theoreticians”, PODS’89)

- **Crux**: Limited recursion on the right

  E.g., Containment of Datalog in UCQ is 2EXPTIME-complete [Chaudhuri+V., 1992]

  – E.g., containment of Datalog in UC2RPQ is 2EXPTIME-complete [Calvanese et al., 2003]

Beyond Graph Databases: Define **GRQ** to be Datalog where *recursion is used only to express transitive closure*.

**Theorem**: **GRQ** containment is 2EXPSPACE-complete [Reutter&Romero&V., 2016]

**Bottom Line**: Full recursion in Datalog is too powerful. Replace in by TC, and you get an expressive but “tractable” fragment – *open question answered!*
From Theory to Practice

**Question:** Any relevance to practice?

**Objections:**

- Isn’t 2EXPSPACE too hard?
  - Do not confuse worst-case complexity with real-life complexity, e.g.: Boolean satisfiability solving, RE containment

- Is query containment really useful for query optimization?
  - The jury is out!
  - Cost-based optimization is dominant!
  - Huge gap between theory and practice!
  - *An inviting research opportunity, especially for declarative networking!*