A Retrospective on Datalog 1.0

Phokion G. Kolaitis

UC Santa Cruz and IBM Research - Almaden

Datalog 2.0
Vienna, September 2012

A Brief History of Datalog

- In the beginning of time, there was E.F. Codd, who gave us relational algebra and relational calculus.
- And then there was SQL.
- In 1979, Aho and Ullman pointed out that SQL cannot express recursive queries.
- In 1982, Chandra and Harel embarked on the study of the expressive power of Datalog.
- Between 1982 and 1995, Datalog “took the field by storm".
- After 1995, interest in Datalog waned for the most part.
- However, Datalog continued to find uses and applications in other areas, such as constraint satisfaction.
- And in recent years, Datalog has made a striking comeback!

Aim and Outline

Aim:

- Highlight and reflect on some themes and results in the study of Datalog.

Outline:

- Complexity and optimization issues in Datalog.
- Tools for analyzing the expressive power of Datalog.
- Datalog and constraint satisfaction.

Disclaimer:

- This talk is not a comprehensive account of Datalog; instead, it is an eclectic mix of topics and results about Datalog that continue to be of relevance.

Datalog: How it all got started

Aho and Ullman - 1979

- Showed that no relational algebra expression can define the Transitive Closure of a binary relation.
  (Shown by logicians earlier; in particular, Fagin – 1975)
- Suggested augmenting relational algebra with fixed-point operators in order to define recursive queries.

Gallaire and Minker - 1978

- Edited a volume with papers from a Symposium on Logic and Databases, held in 1977.

Chandra and Harel - 1982

- Studied the expressive power of logic programs without function symbols on relational databases.

Datalog

Definition

- Datalog $=$ Conjunctive Queries $+$ Recursion

  Function, negation-free, and $\neq$-free logic programs

  Note: The term “Datalog" was coined by David Maier.

Datalog

Definition

- Datalog $=$ Conjunctive Queries $+$ Recursion

  Function, negation-free, and $\neq$-free logic programs

  Note: The term “Datalog" was coined by David Maier.

- A Datalog program is a finite set of rules given by conjunctive queries

  $$T(\overline{x}) \; :- \; S_1(\overline{y}_1), \ldots, S_r(\overline{y}_r).$$

  - Intensional DB predicates (IDBs): Those predicates that occur both in the *heads* and the *bodies* of rules (also known as recursive predicates).
  - Extensional DB predicates (EDBs): All other predicates.

Example (TRANSITIVE CLOSURE Query TC)

$\mathrm{TC}(E) = \{(a, b) : \text{there is a path from } a \text{ to } b \text{ along edges in } E\}$.

A Datalog program for TC:

$$
\left|
\begin{array}{lcl}
S(x, y) & :- & E(x, y) \\
S(x, y) & :- & E(x, z), S(z, y)
\end{array}
\right.
$$

Another Datalog program for TC:

$$
\left|
\begin{array}{lcl}
S(x, y) & :- & E(x, y) \\
S(x, y) & :- & S(x, z), S(z, y)
\end{array}
\right.
$$

- E is the EDB.
- S is the IDB; it defines TC.

Example (Transitive Closure Query TC)

$\text{TC}(E) = \{(a, b) : \text{there is a path from } a \text{ to } b \text{ along edges in } E\}$.

A Datalog program for TC (linear Datalog)

$$\begin{array}{|lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & E(x,z), S(z,y) \end{array}$$

Another Datalog program for TC (non-linear Datalog)

$$\begin{array}{|lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & S(x,z), S(z,y) \end{array}$$

- E is the EDB predicate.
- S is the IDB predicate; it defines TC.

Datalog and 2-Colorability

Example

- Recall that a graph is 2-colorable if and only if it does not contain a cycle of odd length.
- Datalog program for NON 2-COLORABILITY:

$$
\left|\begin{array}{lcl}
O(X,Y) & :- & E(X,Y) \\
O(X,Y) & :- & O(X,Z), E(Z,W), E(W,Y) \\
Q & :- & O(X,X)
\end{array}\right.
$$

- E is the EDB predicate.
- O and Q are the IDB predicates.
- Q defines NON 2-COLORABILITY.

Semantics of Datalog Programs

Declarative Semantics:
Smallest (w.r.t. $\subseteq$) solution to a system of relational algebra equations extracted from the Datalog program.

Procedural Semantics:
“Bottom-up" evaluation of the rules of the Datalog program, starting by assigning $\emptyset$ to every IDB predicate.

Semantics of Datalog Programs

Declarative Semantics:
Smallest (w.r.t. $\subseteq$) solution to a system of relational algebra equations extracted from the Datalog program.

Procedural Semantics:
“Bottom-up" evaluation of the rules of the Datalog program, starting by assigning $\emptyset$ to every IDB predicate.

Fact:
The declarative semantics of a Datalog program coincides with it procedural semantics.

Example: Datalog program for TRANSITIVE CLOSURE:

$$\left| \begin{array}{lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & E(x,z), S(z,y) \end{array} \right.$$

Declarative Semantics: TC is the smallest solution of the relational algebra equation

$$S = E \cup \pi_{1,4}(\sigma_{\$2=\$3}(E \times S)).$$

Procedural Semantics: "Bottom-up" evaluation

$$\left| \begin{array}{lcl} S^0 & = & \emptyset \\ S^{m+1} & = & \{(a,b)) : \exists z(E(a,z) \wedge S^m(z,b))\} \end{array} \right.$$

Fact: The following statements are true:

$$\begin{array}{lcl} S^m & = & \{(a,b) : \text{there is a path of length} \leq m \text{ from } a \text{ to } b\} \\ \mathrm{TC} & = & \bigcup_m S^m \; = \; S^n, \text{where } n \text{ is the number of nodes.} \end{array}$$

Data Complexity of Datalog

Theorem:

- The data complexity of Datalog is PTIME-complete.
- The data complexity of linear Datalog is NLOGSPACE-complete.

Data Complexity of Datalog

Theorem:

- The data complexity of Datalog is PTIME-complete.
- The data complexity of linear Datalog is NLOGSPACE-complete.

Proof:

- Datalog:
  – The “bottom-up" evaluation of a Datalog program converges in polynomially-many steps in the size of the given database.
  – PATH SYSTEMS is expressible in Datalog.
- Linear Datalog:
  – Reduction to TC.
  – TRANSITIVE CLOSURE is expressible in Datalog.

Path Systems and Datalog

Definition (PATH SYSTEMS QUERY)

Given a set A of axioms and a ternary rule of inference R compute the theorems obtained from A using R.

Theorem: Cook - 1974
PATH SYSTEMS is a PTIME-complete problem via log-space reductions.

Fact:
PATH SYSTEMS is definable by the following Datalog program:

$$\left| \begin{array}{lcl} T(x) & :- & A(x) \\ T(x) & :- & R(x,y,z), T(y), T(z) \end{array} \right.$$

The Complexity of Datalog

Query Language	Data Complexity	Combined Complexity
Conjunct. Queries	LOGSPACE	NP-complete
Linear Datalog	NLOGSPACE-compl.	PSPACE-complete
Datalog	PTIME-complete	EXPTIME-complete

Fact:

Since 1999, SQL supports Linear Datalog

Conclusion:

- Datalog can express recursive queries, but this ability is accompanied by a modest increase in data complexity.
- Datalog has tractable data complexity, but not all Datalog queries are efficiently parallelizable.

Datalog Optimization

Fact:

- Datalog optimization has been extensively studied.
- Datalog optimization turned out to be a major challenge.
- Here, we will touch upon just two optimization issues in Datalog:
  1. Boundedness.
  2. Linearizability.

Datalog Boundedness

Definition

Let π be a Datalog program with a single IDB predicate S. We say that π is bounded if there is an integer k such that on every database, the bottom-up evaluation of π converges in at most k steps, that is, $S^k = S^m$, for all $m \geq k$.

Datalog Boundedness

Definition

Let π be a Datalog program with a single IDB predicate S.
We say that π is bounded if there is an integer k such that on every database, the bottom-up evaluation of π converges in at most k steps, that is, $S^k = S^m$, for all $m \geq k$.

Example: The preceding Datalog programs for TRANSITIVE CLOSURE and PATH SYSTEMS are unbounded.

Datalog Boundedness

Definition

Let π be a Datalog program with a single IDB predicate S. We say that π is bounded if there is an integer k such that on every database, the bottom-up evaluation of π converges in at most k steps, that is, $S^k = S^m$, for all $m \geq k$.

Example: The preceding Datalog programs for TRANSITIVE CLOSURE and PATH SYSTEMS are unbounded.

Example: The following Datalog program is bounded ($k = 2$).

$$\left| \begin{array}{lcl} Buys(X, Y) & :- & Likes(X, Y) \\ Buys(X, Y) & :- & Trendy(X), Buys(Z, Y) \end{array} \right.$$

Datalog Boundedness

Note: If a Datalog program π is bounded, then

1. π is equivalent to a finite union of conjunctive queries.
2. The query defined by π is computable in LOGSPACE.

Problem: Design an algorithm for deciding boundedness:
Given a Datalog program π, is it bounded?

Datalog Linearizability

Definition

Let π be a Datalog program with a single IDB predicate S.
We say that π is linearizable if there is a linear Datalog program π^* that is equivalent to π (i.e., π and π^* define the same query).

Datalog Linearizability

Definition

Let π be a Datalog program with a single IDB predicate S.
We say that π is linearizable if there is a linear Datalog program π^* that is equivalent to π (i.e., π and π^* define the same query).

Example: The following Datalog program for TRANSITIVE CLOSURE is linearizable.

$$\begin{array}{|lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & S(x,z), S(z,y) \end{array}$$

Datalog Linearizability

Definition

Let π be a Datalog program with a single IDB predicate S.
We say that π is linearizable if there is a linear Datalog program π^* that is equivalent to π (i.e., π and π^* define the same query).

Example: The following Datalog program for TRANSITIVE CLOSURE is linearizable.

$$\begin{array}{|lcl} S(x,y) & :- & E(x,y) \\ S(x,y) & :- & S(x,z), S(z,y) \end{array}$$

Example: The Datalog program for PATH SYSTEMS is (provably) not linearizable.

$$\begin{array}{|lcl} T(x) & :- & A(x) \\ T(x) & :- & R(x,y,z), T(y), T(z) \end{array}$$

Datalog Linearizability

Note: If a Datalog program π is linearizable, then

1. π is equivalent to a Datalog program that can be evaluated in SQL:1999 and subsequent editions of the SQL standard.
2. The query defined by π is computable in NLOGSPACE.

Problem: Design an algorithm for deciding linearizability: Given a Datalog program π, is it linearizable?

Undecidability in Datalog

Theorem (Gaifman, Mairson, Sagiv, Vardi - 1987)

- There is no algorithm for deciding boundedness.

Undecidability in Datalog

Theorem (Gaifman, Mairson, Sagiv, Vardi - 1987)

- There is no algorithm for deciding boundedness.
- A Rice-type theorem holds for Datalog: If a property P of Datalog programs is *non-trivial*, *semantic*, *stable*, and *contains boundedness*, then P is undecidable.

Undecidability in Datalog

Theorem (Gaifman, Mairson, Sagiv, Vardi - 1987)

- There is no algorithm for deciding boundedness.
- A Rice-type theorem holds for Datalog: If a property P of Datalog programs is *non-trivial*, *semantic*, *stable*, and *contains boundedness*, then P is undecidable.
- In particular, there is no algorithm for deciding linearizability.

Progress Report

- ✓ Complexity and optimization issues in Datalog.
- Tools for analyzing the expressive power of Datalog.
- Datalog and constraint satisfaction.

Analyzing the Expressive Power of Datalog

Question:

- What tools do we have to analyze the expressive power of Datalog?

Answer:

- Preservation under homomorphisms.
- Existential k-pebble games.

Homomorphisms

Definition

Let **A** and **B** be two databases.

- A homomorphism from **A** to **B** is a function $h : \mathrm{adom}(\mathbf{A}) \to \mathrm{adom}(\mathbf{B})$ such that for every relation symbol P and every tuple $(a_1, \ldots, a_n)$ from $\mathrm{adom}(\mathbf{A})$, if $(a_1, \ldots, a_n) \in P^{\mathbf{A}}$, then $(h(a_1), \ldots, h(a_n)) \in P^{\mathbf{B}}$.
- $\mathbf{A} \to \mathbf{B}$ denotes that a homomorphism from **A** to **B** exists.

Homomorphisms

Definition

Let **A** and **B** be two databases.

- A homomorphism from **A** to **B** is a function $h : \mathrm{adom}(\mathbf{A}) \rightarrow \mathrm{adom}(\mathbf{B})$ such that for every relation symbol P and every tuple $(a_1, \ldots, a_n)$ from $\mathrm{adom}(\mathbf{A})$, if $(a_1, \ldots, a_n) \in P^{\mathbf{A}}$, then $(h(a_1), \ldots, h(a_n)) \in P^{\mathbf{B}}$.
- $\mathbf{A} \rightarrow \mathbf{B}$ denotes that a homomorphism from **A** to **B** exists.

Example

- A graph **G** is 2-colorable if and only if $\mathbf{G} \rightarrow \mathbf{K}_2$.
- A graph **G** is 3-colorable if and only if $\mathbf{G} \rightarrow \mathbf{K}_3$.

Preservation under Homomorphisms

Proposition: If a query q is definable by a Datalog program, then q is preserved under homomorphisms, that is, if $\mathbf{A} \models q$ and $\mathbf{A} \rightarrow \mathbf{B}$, then $\mathbf{B} \models q$.

Preservation under Homomorphisms

Proposition: If a query q is definable by a Datalog program, then q is preserved under homomorphisms, that is, if $\mathbf{A} \models q$ and $\mathbf{A} \rightarrow \mathbf{B}$, then $\mathbf{B} \models q$.

Proof:

- Every Datalog program is equivalent to an infinite union of conjunctive queries.
- Every conjunctive query is preserved under homomorphisms.

Preservation under Homomorphisms

Proposition: If a query q is definable by a Datalog program, then q is preserved under homomorphisms, that is, if $\mathbf{A} \models q$ and $\mathbf{A} \rightarrow \mathbf{B}$, then $\mathbf{B} \models q$.

Proof:

- Every Datalog program is equivalent to an infinite union of conjunctive queries.
- Every conjunctive query is preserved under homomorphisms.

Corollary: To show that a query q is not expressible in Datalog, it suffices to show that q is not preserved under homomorphisms.

Preservation under Homomorphisms: Applications

Fact:
None of the following queries is expressible in Datalog:

- "The graph is triangle-free"
  Note that this query is expressible in first-order logic.
- 2-COLORABILITY
  Recall that NON 2-COLORABILITY is expressible in Datalog.
- CONNECTIVITY
- DISCONNECTIVITY
- ...

Analyzing the Expressive Power of Datalog

Question:

- Suppose that q is preserved under homomorphisms, but we believe that q is not expressible in Datalog.

  What tools do we have for confirming this?

- In particular, consider NON 3-COLORABILITY:

  - NON 3-COLORABILITY is preserved under homomorphisms.
  - NON 3-COLORABILITY is coNP-complete.

  How can we show that NON 3-COLORABILITY is not expressible in Datalog?

Datalog, Finite-Variable Logics, and Pebble Games

- Datalog is a fragment of a certain infinitary logic with finitely-many variables.
- The expressive power of this infinitary logic can be captured by existential pebble games.
- Consequently, the expressive power of Datalog can be analyzed using existential pebble games.

Finite-Variable Logics

- An old, but fruitful idea: the number of distinct variables used in formulas is a resource.
- FO^k:
  All first-order formulas with at most k distinct variables.
- If $<$ is a linear order, then

  "there are at least m elements"

  is expressible in FO^2. For example,

  "there are at least 4 *elements"*

  is expressible by

  $$(\exists x)(\exists y)(x < y \wedge (\exists x(y < x \wedge (\exists y)(x < y)))).$$

k-Datalog

Definition

A k-Datalog program is a Datalog program in which each rule $t_0 \ :- \ t_1, \ldots, t_m$ has at most k distinct variables.

Example

- NON 2-COLORABILITY revisited

$$\left|\begin{array}{lll} O(X,Y) & :- & E(X,Y) \\ O(X,Y) & :- & O(X,Z), E(Z,W), E(W,Y) \\ Q & :- & O(X,X) \end{array}\right.$$

- Therefore, NON 2-COLORABILITY is definable in 4-Datalog.
- Exercise: NON 2-COLORABILITY is definable in 3-Datalog.

Finite-Variable Logics and Datalog

Definition (K ... and Vardi - 1995)

If k is a positive integer, then $\exists L^k_{\infty\omega}$ is the collection of all formulas with at most k distinct variables that contains all atomic formulas and is closed under existential quantification, infinitary conjunctions $\bigwedge$, and infinitary disjunctions $\bigvee$.

Finite-Variable Logics and Datalog

Definition (K ... and Vardi - 1995)

If k is a positive integer, then $\exists L^k_{\infty\omega}$ is the collection of all formulas with at most k distinct variables that contains all atomic formulas and is closed under existential quantification, infinitary conjunctions $\bigwedge$, and infinitary disjunctions $\bigvee$.

Theorem: k-Datalog $\subseteq \exists L^k_{\infty\omega}$, for every $k \geq 1$.

Finite-Variable Logics and Datalog

Definition (K ... and Vardi - 1995)

If k is a positive integer, then $\exists L^k_{\infty\omega}$ is the collection of all formulas with at most k distinct variables that contains all atomic formulas and is closed under existential quantification, infinitary conjunctions $\bigwedge$, and infinitary disjunctions $\bigvee$.

Theorem: k-Datalog $\subseteq \exists L^k_{\infty\omega}$, for every $k \geq 1$.

Proof: (By example)

- $P^n(x, y)$: there is a path of length n from x to y.

Finite-Variable Logics and Datalog

Definition (K ... and Vardi - 1995)

If k is a positive integer, then $\exists L^k_{\infty\omega}$ is the collection of all formulas with at most k distinct variables that contains all atomic formulas and is closed under existential quantification, infinitary conjunctions $\bigwedge$, and infinitary disjunctions $\bigvee$.

Theorem: k-Datalog $\subseteq \exists L^k_{\infty\omega}$, for every $k \geq 1$.

Proof: (By example)

- $P^n(x, y)$: there is a path of length n from x to y.
- $P^n(x, y)$ is FO^3-definable:

$$\begin{aligned} P^1(x, y) &\equiv E(x, y) \\ P^{n+1}(x, y) &\equiv \exists z(E(x, z) \wedge \exists x((x = z) \wedge P_n(x, y))). \end{aligned}$$

- Hence, TC $\subseteq \exists L^3_{\infty\omega}$.

Existential k-Pebble Games

Spoiler and Duplicator play on two databases **A** and **B**. Each player uses k pebbles, labeled $1, \ldots, k$. In each move,

- Spoiler places a pebble on or removes a pebble from an element of the active domain **A**.
- Duplicator tries to duplicate the move on **B** using the pebble with the same label.

$$\begin{array}{llllll} \mathbf{A}: & a_1 & a_2 & \ldots & a_l & \\ & \downarrow & \downarrow & \cdots & \downarrow & \\ \mathbf{B}: & b_1 & b_2 & \ldots & b_l & \quad l \leq k \end{array}$$

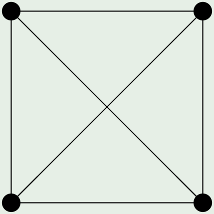
- Spoiler *wins* the $(\exists, k)$-pebble game if at some point the mapping $a_i \mapsto b_i$, $1 \leq i \leq l$, is not a partial homomorphism.
- Duplicator *wins* the $(\exists, k)$-pebble game if the above never happens.

Fact (Cliques of Different Size)

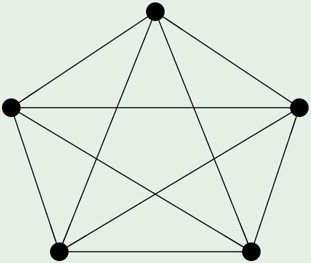
Let $\mathbf{K}_k$ be the k-clique. Then

- Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$.
- Spoiler wins the $(\exists, k)$-pebble game on $\mathbf{K}_k$ and $\mathbf{K}_{k-1}$.

Example



$\mathbf{K}_4$



$\mathbf{K}_5$

Existential Pebble Games and Finite-Variable Logics

Definition

Let k be a positive integer and **A**, **B** be two databases.
A $\preceq_k$ **B** if every $\exists L^k_{\infty\omega}$-sentence that is true on **A** is true on **B**.

Theorem: (K ... and Vardi - 1995)
The following statements are equivalent:

- **A** $\preceq_k$ **B**
- The Duplicator wins the $(\exists, k)$-pebble game on **A** and **B**.

Methodology for Expressibility in Datalog

Corollary: Let q be a Boolean query such that for every $k \geq 1$, there are databases $\mathbf{A}_k$ and $\mathbf{B}_k$ such that

- $\mathbf{A}_k \models q$ and $\mathbf{B}_k \not\models q$.
- The Duplicator wins the $(\exists, k)$-game on $\mathbf{A}$ and $\mathbf{B}$.

Then

- q is not expressible in $\exists L^k_{\infty\omega}$, for any $k \geq 1$.
- In particular, q is not expressible in Datalog.

Methodology for Expressibility in Datalog

Corollary: Let q be a Boolean query such that for every $k \geq 1$, there are databases $\mathbf{A}_k$ and $\mathbf{B}_k$ such that

- $\mathbf{A}_k \models q$ and $\mathbf{B}_k \not\models q$.
- The Duplicator wins the $(\exists, k)$-game on $\mathbf{A}$ and $\mathbf{B}$.

Then

- q is not expressible in $\exists L^k_{\infty\omega}$, for any $k \geq 1$.
- In particular, q is not expressible in Datalog.

Theorem: (Dawar - 1998)
NON 3-COLORABILITY is not expressible in Datalog.

Complexity of the Existential Pebble Game

Problem: Given two databases **A** and **B**, does the Spoiler win the $(\exists, k)$-pebble game on **A** and **B**?

Complexity of the Existential Pebble Game

Problem: Given two databases **A** and **B**, does the Spoiler win the $(\exists, k)$-pebble game on **A** and **B**?

Upper Bound:

- $O(|\mathbf{A}|^{2k}|\mathbf{B}|^{2k}) = O(n^{2k})$, where $n = \max |A|, |B|$.

Complexity of the Existential Pebble Game

Problem: Given two databases **A** and **B**, does the Spoiler win the $(\exists, k)$-pebble game on **A** and **B**?

Upper Bound:

- $O(|\mathbf{A}|^{2k}|\mathbf{B}|^{2k}) = O(n^{2k})$, where $n = \max |A|, |B|$.

Lower Bounds:

Theorem: (K ... and Panttaja – 2003)

- EXPTIME-complete, when k is part of the input.
- PTIME-complete, for each fixed $k \geq 2$.

Complexity of the Existential Pebble Game

Problem: Given two databases **A** and **B**, does the Spoiler win the $(\exists, k)$-pebble game on **A** and **B**?

Upper Bound:

- $O(|\mathbf{A}|^{2k}|\mathbf{B}|^{2k}) = O(n^{2k})$, where $n = \max |A|, |B|$.

Lower Bounds:

Theorem: (K ... and Panttaja – 2003)

- EXPTIME-complete, when k is part of the input.
- PTIME-complete, for each fixed $k \geq 2$.

Theorem: (Berkholz – 2012)

- Not in DTIME$(n^{\frac{k-3}{12}})$, for each fixed $k \geq 15$.

Descriptive Complexity of the Existential Pebble Game

Theorem: (K ... and Vardi - 1998)
For every fixed positive integer k and every fixed database **B**, there is a k-Datalog program that expresses the query:
Given a database **A**, does the Spoiler win the $(\exists, k)$-game on **A** and **B**?

Descriptive Complexity of the Existential Pebble Game

Theorem: (K ... and Vardi - 1998)
For every fixed positive integer k and every fixed database **B**, there is a k-Datalog program that expresses the query:
Given a database **A**, does the Spoiler win the $(\exists, k)$-game on **A** and **B**?

Note:

- This result pinpoints the descriptive complexity of determining the winner in the $(\exists, k)$-pebble game.
- It has been used in the study of Datalog and constraint satisfaction, as we will see next.

Progress Report

- ✓ Complexity and optimization issues in Datalog.
- ✓ Tools for analyzing the expressive power of Datalog.
- Datalog and constraint satisfaction.

The Constraint Satisfaction Problem

Definition (The Constraint Satisfaction Problem - CSP)

Given a set V of variables, a domain D of values, and a set C of constraints on the variables and the values, is there an assignment $s : V \rightarrow D$ so that the constraints in C are satisfied?

The Constraint Satisfaction Problem

Definition (The Constraint Satisfaction Problem - CSP)

Given a set V of variables, a domain D of values, and a set C of constraints on the variables and the values, is there an assignment $s : V \to D$ so that the constraints in C are satisfied?

Examples:

- k-COLORABILITY, for $k \geq 2$.
- k-SAT, for $k \geq 2$

The Constraint Satisfaction Problem

Definition (The Constraint Satisfaction Problem - CSP)

Given a set V of variables, a domain D of values, and a set C of constraints on the variables and the values, is there an assignment $s : V \to D$ so that the constraints in C are satisfied?

Examples:

- k-COLORABILITY, for $k \geq 2$.
- k-SAT, for $k \geq 2$

Fact: (Feder and Vardi – 1993)
CSP can be identified with the HOMOMORPHISM PROBLEM:
Given two databases **A** and **B**, is **A** $\to$ **B**?

The Constraint Satisfaction Problem

Problem: CSP $\equiv$ The Homomorphism Problem:
Given two databases **A** and **B**, is $\mathbf{A} \rightarrow \mathbf{B}$?

Fact: CSP is NP-complete

The Constraint Satisfaction Problem

Problem: CSP $\equiv$ The Homomorphism Problem:
Given two databases **A** and **B**, is $\mathbf{A} \rightarrow \mathbf{B}$?

Fact: CSP is NP-complete

Definition (Non-Uniform CSP)

Let **B** be a fixed database.
$\mathrm{CSP}(\mathbf{B})$ is the following decision problem:
Given a database **A**, is $\mathbf{A} \rightarrow \mathbf{B}$?

Examples:

- $\mathrm{CSP}(\mathbf{K}_2) =$ 2-COLORABILITY (in PTIME)
- $\mathrm{CSP}(\mathbf{K}_3) =$ 3-COLORABILITY (NP-complete)

The Complexity of the Constraint Satisfaction Problem

Dichotomy Conjecture: Feder and Vardi – 1993
For every fixed database **B**, one of the following holds:

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

$\mathrm{CSP}(\mathbf{B})$ $\nearrow$ $\searrow$

NP-complete
$\mathrm{NP} - \mathrm{PTIME}$, not NP-complete
PTIME

The Complexity of the Constraint Satisfaction Problem

Dichotomy Conjecture: Feder and Vardi – 1993
For every fixed database **B**, one of the following holds:

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

$\mathrm{CSP}(\mathbf{B})$ $\nearrow$ $\searrow$

NP-complete
$\mathrm{NP} - \mathrm{PTIME}$, not NP-complete
PTIME

Note:

- The Feder-Vardi Dichotomy Conjecture is still open.
- Extensive interaction between complexity, database theory, logic, and universal algebra towards its resolution.

Question: When is $\mathrm{CSP}(\mathbf{B})$ tractable?

Constraint Satisfaction and Datalog

Question: When is $\mathrm{CSP}(\mathbf{B})$ tractable?

Fact: Feder and Vardi – 1993

- Expressibility in Datalog provides a unifying explanation for many (but not all) tractable cases of $\mathrm{CSP}(\mathbf{B})$.
- More precisely, consider
$$\neg\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \not\rightarrow \mathbf{B}\}.$$
It is often the case that $\mathrm{CSP}(\mathbf{B})$ is in PTIME because $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in Datalog.

Constraint Satisfaction and Datalog

Question: When is $\mathrm{CSP}(\mathbf{B})$ tractable?

Fact: Feder and Vardi – 1993

- Expressibility in Datalog provides a unifying explanation for many (but not all) tractable cases of $\mathrm{CSP}(\mathbf{B})$.
- More precisely, consider
$$\neg\mathrm{CSP}(\mathbf{B}) = \{\mathbf{A} : \mathbf{A} \not\to \mathbf{B}\}.$$
It is often the case that $\mathrm{CSP}(\mathbf{B})$ is in PTIME because $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in Datalog.

Note:

- $\mathrm{CSP}(\mathbf{B})$ is not preserved under homomorphisms.
- $\neg\mathrm{CSP}(\mathbf{B})$ is preserved under homomorphisms.

Constraint Satisfaction and Datalog

Fact: NON 2-COLORABILITY is expressible in Datalog

Constraint Satisfaction and Datalog

Fact: NON 2-COLORABILITY is expressible in Datalog

Fact: HORN 3-UNSAT is expressible in Datalog

- Horn 3-CNF formula φ viewed as a finite structure
  $$\mathbf{A}^{\varphi} = (\{x_1, \ldots, x_n\}, U, P, N), \quad \text{where}$$
  - U is the set of unit clauses;
  - P is the set of clauses of the form $(\neg x \vee \neg y \vee z)$;
  - N is the set of clauses of the form $(\neg x \vee \neg y \vee \neg z)$.
- Datalog program for HORN 3-UNSAT:

$$\left|\begin{array}{lcl} T(z) & :- & U(z) \\ T(z) & :- & P(x,y,z), T(x), T(y) \\ Q & :- & N(x,y,z), T(x), T(y), T(z) \end{array}\right.$$

Unit propagation algorithm for Horn Satisfiability.

Constraint Satisfaction and Datalog

Problems:

- Fix a positive integer k. Can we characterize when $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog?
- Fix a positive integer k. Is there an algorithm for deciding whether, given $\mathbf{B}$, $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog?
- Is there an algorithm for deciding whether, given $\mathbf{B}$, there is some k such that $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog?

Constraint Satisfaction and Datalog

Theorem: (K ... and Vardi – 1998)
Let k be a positive integer and $\mathbf{B}$ a database. The following statements are equivalent:

- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog.
- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in $\exists L^k_{\infty\omega}$.
- $\mathrm{CSP}(\mathbf{B}) =$
  $\{\mathbf{A}$: Duplicator wins the $(\exists, k)$-pebble game on $\mathbf{A}$ and $\mathbf{B}\}$.

Constraint Satisfaction and Datalog

Theorem: (K ... and Vardi – 1998)
Let k be a positive integer and **B** a database. The following statements are equivalent:

- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog.
- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in $\exists L^k_{\infty\omega}$.
- $\mathrm{CSP}(\mathbf{B}) =$
  $\{\mathbf{A}$: Duplicator wins the $(\exists, k)$-pebble game on **A** and **B**$\}$.

Note:

- In general, k-Datalog $\subset \exists L^k_{\infty\omega}$.
- Single *canonical* PTIME-algorithm for all $\mathrm{CSP}(\mathbf{B})$'s that are expressible in k-Datalog, for fixed k, namely:
  Determine the winner in the $(\exists, k)$-pebble game.

CSP, Datalog, and Universal Algebra

Theorem: (Barto and Kozik – 2009)

- Expressibility of $\neg\mathrm{CSP}(\mathbf{B})$ in Datalog can be characterized in terms of *tame congruence theory* in universal algebra.
- There is an EXPTIME-algorithm for the following problem: Given **B**, is there some k such that $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog?
- There is a PTIME-algorithm for the following problem: Given a *core* **B**, is there some k such that $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog?

Note: Deep and *a priori* unexpected connection between constraint satisfaction, Datalog, and universal algebra.

CSP and the Collapse of the k-Datalog Hierarchy

Fact:
k-Datalog is strictly more expressive than k'-Datalog, for $k > k'$.

CSP and the Collapse of the k-Datalog Hierarchy

Fact:
k-Datalog is strictly more expressive than k'-Datalog, for $k > k'$.

Theorem: (Barto – 2012; implicit in Barto and Kozik – 2009)
Let **B** be a fixed database over a schema of maximum arity r.
The following statements are equivalent:

- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in k-Datalog, for some k.
- $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in $\max(3, r)$-Datalog.

Note: This is a theorem about logic whose only known proof is via universal algebra!

CSP and Linear Datalog

Fact:
If $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in linear Datalog, then $\mathrm{CSP}(\mathbf{B})$ is in NLOGSPACE.

CSP and Linear Datalog

Fact:
If $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in linear Datalog, then $\mathrm{CSP}(\mathbf{B})$ is in NLOGSPACE.

Open Problems:

- Is there a database **B** such that $\mathrm{CSP}(\mathbf{B})$ is in NLOGSPACE, but $\neg\mathrm{CSP}(\mathbf{B})$ is not expressible in linear Datalog?
- Is there an algorithm for deciding whether, given **B**, $\neg\mathrm{CSP}(\mathbf{B})$ is expressible in linear Datalog?

Note: Universal algebra methods have been applied towards these problems and partial results have been recently obtained.

Concluding Remarks

- The study of Datalog has been a meeting point of database theory, computational complexity, logic, universal algebra, and constraint satisfaction. It has resulted into a fruitful interaction between these areas.
- One can only hope that the next thirty years of Datalog will be as fruitful as the first thirty.