Datalog: A perspective and the potential

Yuri Gurevich

Microsoft Research, Redmond, Washington, USA

What we see depends mainly on what we look for.
—John Lubbock

Abstract. Our main goal is to put Datalog into a proper logic perspective. It may be too early to put Datalog into a proper perspective from the point of view of applications; nevertheless we discuss why Datalog pops up so often in applications.

1 Introduction

Throughout our career, we came across Datalog many times, directly or indirectly. This exposition is, in a sense, a summary of our experience. Here we are interested in proper perspectives on Datalog from the point of view of logic and applications. The exposition contains no new hard technical results.

A short §2 on preliminaries seeks to fix the terminology and make this exposition more self-contained. In §3, we recall the notion of global relations; first-order and second-order formulas are, semantically, global relations. According to the standard model-theoretic semantics of Datalog, known also as the fixed-point semantics, Datalog queries are global relations as well. We also recall the proof-theoretic semantics of Datalog queries, and we formulate how the two semantics of Datalog are related.

The goal of the following §4–7 is to put Datalog into a proper logic perspective. The global-relation view allows us to compare the expressive power of Datalog with that of more traditional logics. Whether we speak about all structures or only finite structures, Datalog has only trivial intersection with first-order logic (§4) and constitutes only a tiny fragment of second-order logic (§5). There is, however, a more traditional logic whose expressive power is exactly that of Datalog; it is existential fixed-point logic (§7). The equiexpressivity result is rather robust.

In applications, there is a growing interest in rule-based systems, and Datalog emerges as a convenient and popular basis for such systems. One instructive example is "Dedalus: Datalog in Time and Space" [3]. In §8, we illustrate the use and limitations of Datalog for policy/trust management, and then we describe an extension of Datalog, called primal infon logic, that overcomes indicated limitations while preserving the feasibility of Datalog. To this end, Datalog is viewed as a logic calculus without axioms. Primal infon logic extends that calculus with axioms and more inference rules.

Is Datalog just a fleeting fashion or is there something objective in its coming up again and again in different applications? Following a recent article [10], our final section §9 gives an argument for the latter. It turns out that standard logic systems (and even many non-logic systems) reduce to Datalog. While many of these reductions are infeasible, some of them are rather practical and allow one to exploit well-optimized Datalog tools.

Acknowledgment

Many thanks to Andreas Blass, Carlos Cotrini and Phokion Kolaitis for useful discussions.

2 Preliminaries

By default, first-order formulas are without equality or function symbols of positive arity, and a term is a variable or constant.

A (pure) Datalog program is built from atomic formulas of first-order logic. The relation symbols of the program split into *extensional* and *intensional*; accordingly the atomic formulas are extensional or intensional. The program itself is a finite set of facts and rules. Facts are extensional atomic formulas, and rules have the form

$$\beta_0 \ :- \ \alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_\ell \tag{1}$$

where all formulas α_i are atomic and extensional, formulas β_j are atomic and intensional, and k, ℓ may be zero. The *implication form* of (1) is a formula

$$(\alpha_1 \wedge \cdots \wedge \alpha_k \wedge \beta_1 \wedge \cdots \wedge \beta_\ell) \to \beta_0, \tag{2}$$

and the *closed form* of (1) is a sentence

$$\forall \bar{x} \big((\alpha_1 \wedge \cdots \wedge \alpha_k \wedge \beta_1 \wedge \cdots \wedge \beta_\ell) \to \beta_0 \big) \tag{3}$$

where $\bar{x}$ comprises the individual variables of (1). Similarly, the *closed form* of a fact $\alpha(\bar{x})$ is the sentence $\forall \bar{x}\, \alpha(\bar{x})$.

A Datalog query Q is a pair (Π, γ) where Π is a Datalog program and γ an intensional atomic formula. The extensional vocabulary of Π (resp. Q) comprises the constants and extensional relation symbols in Π (resp. Q). The number of distinct variables in γ is the arity of Q. Nullary queries are also known as ground queries.

Example 1. Let Π_1 be the Datalog program

$$\begin{aligned} E(x, 1) & \\ E(2, 3) & \\ T(3, x) & :- \\ T(x, y) & :- E(x, y) \\ T(x, y) & :- T(x, z), T(z, y) \end{aligned}$$

with two facts and three rules. Relation E is extensional, and relation T is intensional. The extensional vocabulary of query $(\Pi_1, T(1,4))$ consists of the constants $1, 2, 3, 4$ and the relation symbol E. □

Let Π be a Datalog program, Υ the extensional vocabulary of Π and Υ' the result of adding to Υ the intensional symbols of Π. The program Π gives rise to an operator O that, given any Υ'-structure B, does the following. For every rule (1) of Π and every instantiation

$$\xi\beta_0 \ :- \ \xi\alpha_1, \ldots, \xi\alpha_k, \xi\beta_1, \ldots, \xi\beta_\ell$$

of the variables of the rule with elements of B such that the ground atomic formulas $\xi\alpha_1, \ldots, \xi\alpha_k, \xi\beta_1, \ldots, \xi\beta_\ell$ hold in B, the operator O sets $\xi\beta_0$ true (unless it was true already in B). The result is an Υ'-structure $O(B)$ which is like B except that the intensional relations might have grown.

Given any Υ-structure A modeling (the closed form of) the facts of Π, let A_0 be the Υ'-expansion of A where all the intensional relations are empty, $A_{n+1} = O(A_n)$, and A^* be the limit of structures A_n so that, for every intensional relation symbol R, the interpretation of R in A^* is the union of its interpretations in structures A_n. The intensional relations of A^* are the smallest intensional relations closed under the rules of Π over A. In other words, they are the smallest intensional relations that make the closed forms of the rules true.

Definition 2. A query $(\Pi, \gamma(\bar{x}))$ is *bounded* if there exists a number n such that, for every structure A, we have

$$\{\bar{a} : \ A_n \models \gamma(\bar{a})\} = \{\bar{a} : \ A^* \models \gamma(\bar{a})\}. \tag{4}$$

The query is bounded on a class $\mathcal{C}$ of structures if there exists n such that (4) holds on all structures in $\mathcal{C}$. □

3 Proof-theoretic and model-theoretic semantics of Datalog programs and queries

There are two different semantics of Datalog queries in the literature. It may be useful to clarify what they are and how they are related.

3.1 Proof-theoretic semantics

View a given Datalog program Π as a deductive system. The axioms of Π are its facts. Each rule (1) of Π gives rise to an inference rule

$$\frac{\xi\alpha_1, \ \ldots, \ \xi\alpha_k, \ \xi\beta_1, \ \ldots, \ \xi\beta_\ell}{\xi\beta_0} \tag{5}$$

where ξ is an arbitrary substitution (of variables with terms). And there is one additional inference rule, the substitution rule

$$\frac{\varphi}{\xi\varphi} \tag{6}$$

where ξ is again an arbitrary substitution.

A nullary query (Π, γ) is an assertion that γ is Π-deducible. It is easy to see that, in Example 1, $T(1,4)$ is not Π_1-deducible.

Lemma 3. *Let $(\Pi, \gamma(\bar{x}))$ be a Datalog query, H a set of atomic formulas, and $\gamma(\bar{c})$ the result of replacing the variables $\bar{x}$ with fresh constants $\bar{c}$. Then*

$$H \vdash_\Pi \gamma(\bar{x}) \iff H \vdash_\Pi \gamma(\bar{c}).$$

Proof. $\Longrightarrow$. Use the substitution rule.
$\Longleftarrow$. Given a derivation of $\gamma(\bar{c})$, replace the constants $\bar{c}$ with fresh variables $\bar{y}$ and then use the substitution rule. □

3.2 Model-theoretical semantics

Definition 4 ([14]). An r-ary *(abstract) global relation* $\mathcal{R}$ of vocabulary Υ associates with any given Υ-structure A an r-ary relation $\mathcal{R}A$ on (the base set of) A in such a way that

$$\mathcal{R}\eta A = \eta\mathcal{R}A$$

for every isomorphism η from A to another Υ-structure. □

Any first-order or second-order formula φ is, semantically, a global relation $A \models \varphi$. Model-theoretically, a Datalog query $Q = (\Pi, \gamma(\bar{x}))$ is also a global relation $\mathcal{R}$; the vocabulary of $\mathcal{R}$ is the extensional vocabulary of Q, and the arity of $\mathcal{R}$ is the number of distinct variables in γ. A relation $\mathcal{R}A(\bar{x})$ asserts that $A \models \gamma(\bar{x})$ if A models the (closed form of the) facts of Π and if the intensional relations over A are as in the definition of A^* in §2.

We illustrate that global relation $\mathcal{R}$ on the example of a binary query $(\Pi_1, T(x,y))$ where Π_1 is the the program of Example 1. Let G be an arbitrary directed graph (V, E) with distinguished (and not necessarily distinct) elements $1, 2, 3$. If $E(2,3)$ or $\forall x\, E(x,1)$ fails in G, then relation $\mathcal{R}G(x,y)$ is universally true. If $E(2,3)$ and $\forall x\, E(x,1)$ hold in G, consider the closed forms

$$\begin{aligned}
\rho_1 &= \forall x\, T(3,x) \\
\rho_2 &= \forall x,y\, (E(x,y) \to T(x,y)) \\
\rho_3 &= \forall x,y,z\, ((T(x,z) \wedge T(z,y)) \to T(x,y))
\end{aligned}$$

of the rules of Π_1. Interpret T as the least relation on V such that the expansion $G^* = (V, E, T)$ satisfies ρ_1, ρ_2 and ρ_3. The relation $\mathcal{R}G(x,y)$ is $G^* \models T(x,y)$.

3.3 Relating the two semantics

Theorem 5. *Let $\mathcal{R}$ be the global relation of a Datalog query $(\Pi, \gamma(\bar{x}))$, F the collection of the facts of Π and H a set of atomic formulas with relation symbols different from the intensional symbols of Π. The following claims are equivalent.*

1. $H \vdash_\Pi \gamma(\bar{x})$.
2. $\mathcal{R}A(\bar{x})$ *is universally true in every structure A satisfying $F \cup H$.*

Proof. Without loss of generality, we may assume that $\gamma(\bar{x})$ is ground. Indeed, instantiate the variables $\bar{x}$ with fresh constants $\bar{c}$. Now use Lemma 3 and the obvious fact that $\mathcal{R}A(\bar{x})$ is universally true in a structure A if and only if $\mathcal{R}A(\bar{c})$ holds in every expansion of A with constants $\bar{c}$ (and the same base set).

$1 \to 2$ is obvious.
$2 \to 1$. We suppose that claim 1 fails and prove that claim 2 fails as well. Let Υ be the extensional vocabulary of the query extended with that of H. Without loss of generality, Υ contains at least one constant. Consider the Υ-structure A on the Υ-constants where an extensional ground atomic formula α holds if and only if it is an instantiation of a hypothesis or fact. It suffices to prove that $\mathcal{R}A$ is false. Obviously the facts and hypotheses are universally true in A.

Let A' be the expansion of A with the intensional relations of Π where an intensional ground atomic formula β holds if and only if it is Π-deducible from H. This instantiates intensional variables to the least values satisfying the closed forms of the rules of Π. Taking into account that γ is not Π-deducible from H, it follows that $\mathcal{R}A$ fails. □

One case of interest is $H = \emptyset$. Another one is where H is the positive diagram $\Delta^+(A)$ of a structure A such that A models F and every element of A is distinguished (a constant). Here $\Delta^+(A)$ is the set of all ground atomic formulas in the vocabulary of A that are true in A.

4 Datalog and first-order logic

There is a bit of confusion in the literature about the relation of Datalog and first-order logic. "Query evaluation with Datalog is based on first order logic" [23]. "Datalog is declarative and is a subset of first-order logic" [19]. In fact, Datalog is quite different from first-order logic. Datalog is all about recursion, and first-order logic does not have any recursion (though recursion is available in some first-order theories, e.g. arithmetic). We say that a Datalog query and a first-order formula are equivalent if their global relations coincide. More generally, the query and formula are equivalent on a class $\mathcal{C}$ of structures if their global relations coincide on $\mathcal{C}$.

Theorem 6. *If a Datalog query is equivalent to a first-order formula then the query is bounded and equivalent to a positive existential first-order formula.*

Theorem 6 is a straightforward consequence of the compactness theorem [2, Theorem 5]. Unfortunately the compactness argument involves infinite structures of little relevance to Datalog applications. The finite version of Theorem 6 was more challenging.

Theorem 7 (Ajtai-Gurevich). *If a Datalog query is equivalent to a first-order formula on finite structures then, on finite structures, the query is bounded and equivalent to a positive existential first-order formula.*

Theorem 7 is fragile [2, §10] as far as extensions of Datalog are concerned. It was generalized in [4] to some classes of finite structures. Later Benjamin Rossman proved the powerful Homomorphism Preservation Theorem [21] that implies Theorem 7.

5 Datalog and second-order logic

Theorem 8. *Every Datalog query (Π, γ) is equivalent to a second-order formula Φ of the form $\forall \bar{X} \exists \bar{y} \varphi$ where $\bar{X}$ is a sequence of relation variables, $\bar{y}$ is a sequence of individual variables and φ is quantifier-free.*

Again, the equivalence of a query and formula means that their global relations coincide. The form $\forall \bar{X} \exists \bar{y} \varphi$ is known as the strict $\forall^1_1$ form where "strict" refers to the fact that the first-order part is existential. Strict $\forall^1_1$ formulas were studied by logicians long before Datalog was introduced [20].

For illustration consider Datalog query $(\Pi_1, T(a, b)$ where Π_1 is the program of Example 1 and a, b are fresh constants. Let F be the formula

$$(\forall x E(1, x)) \wedge E(2, 3)$$

reflecting the facts of Π_1 and let ρ_1, ρ_2 and ρ_3 be the closed forms of the rules of Π_1, as in §3. Then the desired Φ is (the strict $\forall^1_1$ formula equivalent to) the formula

$$\neg F \vee \forall T\big((\rho_1 \wedge \rho_2 \wedge \rho_3) \rightarrow T(a, b)\big)$$

The converse of Theorem 8 is not true: non-3-colorability is expressible by a strict $\forall^1_1$ formula while it cannot be expressed by any Datalog query unless P = NP [6]. The converse can be obtained by severely restricting the form of the quantifier-free formula φ.

6 Liberal Datalog

The version of Datalog considered above is known as pure Datalog. It has been generalized in numerous ways. In particular, Constraint Datalog is popular; see for example [19] and references there. One very different generalization started with article [14] where we defined and studied inflationary fixed points. Abiteboul and Vianu used inflationary-fixed-point semantics to define an elegant version of Datalog with negations [1] that was popularized by Ullman [22] and used e.g. in [3]. One simple and most natural liberalization of pure Datalog is this:

(a) Make extensional atomic formulas negatable, so that the facts of a program are extensional atomic formulas or their negations, and rules have the form (1) where $\alpha_1, \ldots, \alpha_k$ are extensional atomic formulas or their negations and formulas β_j are atomic and intensional.

While the positivity of intensional formulas is essential for the least-fixed-point construction, the requirement that extensional formulas be positive has not been essential above. The whole §3 remains valid under liberalization (a). Furthermore Theorem 5 and its proof remain valid if "H is a set of atomic formulas" is replaced with "H is a set of atomic formulas or their negations." A new special case of interest is where H is the diagram $\Delta(A)$ of a structure A on constants that models F. Here $\Delta(A)$ is the set of all ground atomic formulas and their negations in the vocabulary of A that are true in A.

A further liberalization of Datalog was introduced in [16], by the name Liberal Datalog, and was studied in [8]. In addition to (a), there are two additional liberalizations in Liberal Datalog.

(b) The extensional vocabulary of a program may contain function symbols of any arity. Intensional formulas may contain extensional function symbols.
(c) Equality has its usual meaning and may occur in programs as an extensional relation symbol.

Model-theoretically, liberal Datalog queries are global relations. Theorem 8 remains valid for Liberal Datalog queries [6, Theorem 5].

7 Datalog, Liberal Datalog, and existential fixed-point logic

We have seen that Datalog has a trivial intersection with first-order logic and constitutes only a sliver of second-order logic. Existential fixed-point logic (EFPL) was introduced as the right logic to formulate preconditions and post-conditions of Hoare logic [6]. The same authors continued to investigate EFPL in [7,8,9].

Theorem 9 ([8]). *Every global relation expressible by an EFPL formula is expressible by a Liberal Datalog query, and the other way round.*

`Q:` If you want a logic with the expressivity of Liberal Datalog, why not declare Liberal Datalog a logic in its own right?

`A:` In traditional logics, like first-order logic or second-order logic, logic operators are explicit and can be nested. EFPL is traditional from that point of view. While in Datalog, pure or liberal, the fixed-point operation is implicit and can't be nested, in EFPL it is explicit and can be nested.

For the purpose of the following theorem, equality is considered part of logic and therefore is not counted in the definition of the vocabulary of an EFPL formula.

Theorem 10 (Blass-Gurevich). *Every global relation expressible by an EFPL formula without negations or function symbols of positive arity is expressible by a pure Datalog query, and the other way round.*

We explain how to prove Theorem 10 given the proof of Theorem 9. In the proof of Theorem 9, given an EFPL formula φ, we construct a Datalog query Q with the same global relation, and the other way round. The vocabulary of φ coincides with the extensional vocabulary of Q; if one of them has no function symbols of positive arity, neither does the other. If the given φ has no negation then the constructed Q has no negation, and the other way round. However, the construction of φ from Q makes use of equality. Equality is legal in EFPL but pure Datalog does not have it. The problem arises how to deal with the equality of φ in the construction of Q from φ. Pure Datalog does not have equality as an extensional relation. But, since we need only positive occurrences of equality, we can compute the equality as an intensional relation:

$$E(x, x) \ :-$$

The intensional relation E represents the equality of φ in the construction of Q.

8 Datalog, policies and primal logic

The advent of cloud computing forces us to be more careful with policies and trust. Numerous policies, that might have been implicit and vague in the world of brick and mortar, need be explicit, precise and automated in the cloud. Many policy rules are expressible in Datalog.

```
X can read File 13  :-  Alice owns File 13,
                        Alice and X are friends.
```

But there are common policy rules that are not expressible in Datalog, primarily because they involve quotations and trust implications.

```
X can read File 13 :-   Alice owns File 13,
                        Alice said Friends(A,X),
                        Alice is trusted on Friends(A,X).
```

where `Friends(A,X)` is a more formal version of `Alice and X are friends` and `Alice is trusted on Friends(A,X)` abbreviates the implication

$$(\texttt{Alice said Friends}(\texttt{A}, \texttt{X})) \rightarrow \texttt{Friends}(\texttt{A}, \texttt{X}).$$

Primal infon logic, introduced in [17], is a proof-theoretic extension of Datalog that allows one to use quotations and nested implications. (Infons are statements treated as pieces of information.)

In the rest of this section, formulas are by default quantifier-free first-order formulas without equality or function symbols of positive arity. Datalog rules (1) will be viewed as implications (2) so that a Datalog program is a finite set of formulas.

8.1 Proof-theoretic semantics of Datalog

As we have seen in §3, Datalog programs can be viewed as logic calculi, but there is a broader proof-theoretic view of Datalog according to which Datalog itself is a logic calculus. The logic calculus DL of pure Datalog has no axioms and just three inference rules. One of them is the substitution rule (6). The other two are the the following conjunction introduction rule and implication elimination rule:

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad\qquad \frac{\varphi \quad \varphi \to \psi}{\psi}.$$

We write $H \vdash_{\mathrm{DL}} \varphi$ to indicate that hypotheses H entail formula φ in DL.

Theorem 11. *For any Datalog query (Π, γ), we have* $\vdash_\Pi \gamma \iff \Pi \vdash_{\mathrm{DL}} \gamma$.

Proof. $\Longrightarrow$. To simulate a rule of Π, use conjunction introduction to derive the body of the rule, and then use implication elimination to derive the head.
$\Longleftarrow$. Check by induction on the derivation length that Π entails only Π-derivable atomic formulas and their conjunctions.

The entailment problem for the ground fragment of DL, obtained from DL by removing the substitution rule, is solvable in linear time [13].

8.2 Primal infon logic

The quote-free fragment of primal infon logic is obtained from DL by adding an axiom $\top$ and the following conjunction elimination rules and implication introduction rule:

$$\frac{\varphi \wedge \psi}{\varphi} \qquad \frac{\varphi \wedge \psi}{\psi} \qquad\qquad \frac{\varphi}{\psi \to \varphi}.$$

To form quotations, primal infon logic has countably infinite lists of variables and constants of type Principal. The logic uses unary connectives p `said`[1] where p is a term of type Principal. A quotation prefix is a string of the form

$$p_1 \texttt{ said } p_2 \texttt{ said } \ldots\ p_k \texttt{ said}.$$

Primal infon logic is given by the following logic calculus where `pref` ranges over quotation prefixes.

[1] Originally primal infon logic had two kinds of quotations p `said` φ and p `implied` φ but later the second kind was removed.

Axioms $\quad \texttt{pref}\ \top$

Inference rules

$$\frac{\texttt{pref}\,(x \wedge y)}{\texttt{pref}\,x} \qquad \frac{\texttt{pref}\,(x \wedge y)}{\texttt{pref}\,y}$$

$$\frac{\texttt{pref}\,x \qquad \texttt{pref}\,y}{\texttt{pref}\,(x \wedge y)}$$

$$\frac{\texttt{pref}\,x \qquad \texttt{pref}\,(x \to y)}{\texttt{pref}\,y}$$

$$\frac{\texttt{pref}\,y}{\texttt{pref}\,(x \to y)}$$

The entailment problem for the ground fragment of primal logic, obtained from primal logic by removing the substitution rule, is solvable in linear time [15,12]. This is important because, while policy rules typically have variables, deduction often deals with fully instantiated cases. An article [12] is being written to become a standard initial reference for primal infon logic.

9 It all reduces to Datalog

In a way, pure Datalog reflects the essence of deductive systems. By default, this section follows [10].

Definition 12. A *Hilbertian system* (or an *abstract Hilbertian deductive system*) is given by a set F of so-called *formulas*, a subset $\mathrm{Ax} \subseteq F$ of so-called *axioms*, and a set Ru of so-called *rules of inference* $\langle P, \alpha \rangle$ where P is a finite subset of F and $\gamma \in F$.

`Q:` Why these "so-called"?

`A:` There are Hilbertian systems that do not look at all like logics. For example, let F be the set of edges of a fixed digraph, $\mathrm{Ax} = \emptyset$, and Ru comprise the pairs $\langle \{(a,b),(b,c)\},(a,c)\rangle$. Yet we'll call the elements of F, Ax and Ru formulas, axioms and rules respectively.

Theorem 13. *For every Hilbertian system S there is a (possibly infinite) Datalog program Π such that S-formulas are propositional symbols of Π and*

$$H \vdash_S \gamma \iff H \vdash_\Pi \gamma.$$

Definition 14. A Hilbertian system is *substitutional* if:

1. Formulas are certain finite strings in a specified alphabet.

2. The alphabet includes a countably infinite set of so-called *variables*, and some (possibly none) of the non-variable symbols are so-called *constants*. The variables and constants are *terms*.
3. If α is a formula then the result $\xi\alpha$ of replacing in α distinct variables x by terms $\xi(x)$ respectively is also a formula, and $\langle\{\alpha\}, \xi\alpha\rangle$ is a rule of inference. Such rules are *substitution rules*.
4. If $\langle\{\alpha_1, \ldots, \alpha_n\}, \beta\rangle$ is a rule of inference that is not a substitution rule, then $\langle\{\xi\alpha_1, \ldots, \xi\alpha_n\}, \xi\beta\rangle$ is also a rule of inference, for any substitution ξ. □

The requirement 4 is often superfluous. For example, the inference rules of primal infon logic are closed under substitutions. Here is a simple example when the requirement is essential. Consider a deductive system with axiom $P(1)$, the substitution rule and a rule $\dfrac{P(x)}{Q(x)}$. One may expect to derive $Q(1)$, but it is not derivable in the system.

Theorem 15. *For every substitutional Hilbertian system S there is a (possibly infinite) Datalog program Π such that Π treats any S-formula α with k distinct variables as a relation symbol of arity k, and*

$$H \vdash_S \gamma \iff H \vdash_\Pi \gamma.$$

Theorem 16. *There is an algorithm that converts any instance of the derivability problem for primal infon logic into an instance of the derivability problem for pure Datalog, with the same answer.*

A more practical algorithm for the same purpose is constructed in [5].

References

1. Serge Abiteboul and Victor Vianu, "Datalog Extensions for Database Queries and Updates," J. of Computer and System Sciences 43 (1991), 62–124.
2. Miklos Ajtai and Yuri Gurevich, "Datalog vs First-Order Logic," J. of Computer and System Sciences 49:3 (1994), 562–588.
3. Peter Alvaro, William Marczak, Neil Conway, Joseph M. Hellerstein, David Maier and Russell C Sears, "Dedalus: Datalog in Time and Space," EECS Dept, University of California, Berkeley Technical Report No. UCB/EECS-2009-173, December 16, 2009.
4. Albert Atserias, Anuj Dawar and Phokion Kolaitis, "On Preservation under Homomorphisms and Unions of Conjunctive Queries," JACM 53:2 (2006), 208-237.
5. Nikolaj Bjørner, Guido de Caso and Yuri Gurevich, "From Primal Infon Logic with Individual Variables to Datalog," Springer Lecture Notes in Computer Science 7265 (1012), 72–86.
6. Andreas Blass and Yuri Gurevich, "Existential Fixed-Point Logic," Springer Lecture Notes in Computer Science 270 (1987) 20–36.
7. Andreas Blass and Yuri Gurevich, "The Underlying Logic of Hoare Logic," Bull. EATCS 70 (2000), 82–110, also in *Current Trends in Theoretical Computer Science*, World Scientific (2001), 409–436.

8. Andreas Blass and Yuri Gurevich, "Two Forms of One Useful Logic: Existential Fixed Point Logic and Liberal Datalog," Bull. EATCS 95 (2008), 164–182.
9. Andreas Blass and Yuri Gurevich, "One Useful Logic That Defines Its Own Truth," Springer Lecture Notes in Computer Science 5162 (2008), pages 1–15.
10. Andreas Blass and Yuri Gurevich, "Hilbertian Deductive Systems, Infon Logic, and Datalog," Microsoft Research Technical Report MSR-TR-2011-81 (June 2011), to appear in Postproceeding from FCT'2011 in Information and Computation.
11. Stephen A. Cook, "Soundness and Completeness of an Axiom System for Program Verification," SIAM J. Computing 7 (1978), 70–90.
12. Carlos Cotrini and Yuri Gurevich, "Revisiting Primal Infon Logic," in preparation.
13. Evgeny Dantzin, Thomas Eiter, Georg Gottlob and Andrei Voronkov, "Complexity and Expressive Power of Logic Programming," *ACM Computing Sureveys* 33:3 (2001), 374–425.
14. Yuri Gurevich, "Toward Logic Tailored for Computational Complexity," Springer Lecture Notes in Math. 1104 (1984), 175–216.
15. Yuri Gurevich, "Two Notes on Propositional Primal Logic," Microsoft Research Tech. Report MSR-TR-2011-70, May 2011
16. Yuri Gurevich and Itay Neeman, "DKAL: Distributed-Knowledge Authorization Language," 21st IEEE Computer Security Foundations Symposium (CSF 2008), 149-162.
17. Yuri Gurevich and Itay Neeman, 'DKAL 2 — A Simplified and Improved Authorization Language," Microsoft Research Tech. Report MSR-TR-2009-11.
18. Yuri Gurevich and Saharon Shelah, "Fixed-Point Extensions of First-Order Logic," Annals of Pure and Applied Logic 32 (1986), 265–280.
19. Ninghui Li and John C. Mitchell, "Datalog with Constraints: A Foundation for Trust Management Languages," in V. Dahl and P. Wadler (Eds.), PADL 2003, Springer LNCS 2562 (2003), 58-73.
20. Anatoly I. Maltsev, "Model Correspondences," Izv. Akad. Nauk SSSR. Ser. Mat. 23 (1959), 313–336 (Russian).
21. Benjamin Rossman, "Homomorphism Preservation Theorems," JACM 55:3 (2008), Article No. 15.
22. Jeffrey D. Ullman, "Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies," W. H. Freeman & Co., New York, NY, USA, 1990.
23. Wikipedia, "Datalog," http://en.wikipedia.org/wiki/Datalog (viewed on June 14, 2012).