The Semiring Framework for Database Provenance

Todd J. Green
LogicBlox Inc.
todd.green@logicblox.com

Val Tannen
Computer and Information Science Department
University of Pennsylvania
val@cis.upenn.edu

This paper does not offer a survey of the semiring framework. Rather, it is intended to bring forward, with the benefit of 20/20 (well, at least 15/20) hindsight, a couple of insights that were not emphasized in the papers that constitute the core of the framework's development. These insights do not even pertain to all the technical aspects of the framework, leaving out, most notably, Datalog. We have also tried to collect a reasonably large set of relevant references. Most of all, we hope to make the reader intrigued enough to follow up with additional reading and perhaps examine further research questions.

Data provenance.

Imagine a computational process that uses a complex input consisting of multiple items. The granularity and nature of "input item" can vary significantly. It can be a single tuple, a database table, or a whole database. It can a spreadsheet describing an experiment, a laboratory notebook entry, or another form of capturing annotation by humans in software. It can also be a file, or a storage system component. It can be a parameter used by a module in a scientific workflow. It can also be a configuration rule used in software-defined routing or in a complex network protocol. Or it can be a configuration decision made by a distributed computation scheduler (think map-reduce). *Provenance analysis* allows us to understand how these different input items affect the output of the computation. When done appropriately, such analysis can be further used, for example,

A1: to figure out how much to trust the output, assuming that we may trust some input items more than others;

A2: to minimize the cost of obtaining the output, assuming that one has to pay for the input items;

A3: to figure out the clearance level required for accessing the output, assuming that we know the clearance levels for the input items;

A4: to compute the probabilistic distribution of the output, assuming that we know the distributions of the input items;

A5: to figure out if the output might change (and therefore whether output *maintenance* is necessary) when certain input items change; or

A6: to track back and find the input items at fault, assuming the output is somehow wrong.

We shall have occasion below to refer to the applications A1-A6 just listed. In practice, a computational process will produce a collection of output items and the applications above become more interesting when we realize that we may get different analyses for each output item.

Now, observe that these applications should not rely on just a trace, or a log, of a specific execution of the computational process itself. [1] Some approach to applications A3 and A5 could probably be devised using just execution traces, but we should still worry whether repeating the execution with the same input items but with a differently optimized execution platform will produce the same provenance. At the same time, we cannot simply peg provenance as a kind of static analysis of the underlying program since it will also depend on the input items themselves. This dual static-dynamic nature makes provenance analysis particularly interesting, especially so for database computations.

Database provenance.

Provenance analysis for queries, views, database ETL tools, and schema mappings takes advantage of their declarative nature. An early approach for RDBMS [18, 19], let's call it *which-provenance*, produces a set of tuple ids that answer the question *" which tuples in the input database contributed to obtaining a particular output tuple?"*. When an output tuple can be derived in multiple ways, which-provenance collects the contributing tuples in each of these ways but then puts them all in the same set. This yields a (workable, but not completely satisfactory) solution for applications A3, A5, and A6, however it does not help with any of the other applications. This work was followed by a seminal contribution: the *why-provenance* of [14] which collects contributing input tuples separately for each derivation of an output tuple. The result is a set of *witnesses*, each witness being a set of input tuples. We illustrate the difference between which- and why-provenance in Figure 1 where we

[1] Although in many existing approaches provenance collection consists of just adding entries to a log.

PODS'17, May 14 - 19, 2017, Chicago, IL, USA

ISBN 978-1-4503-4198-1/17/05…15.00

DOI: http://dx.doi.org/10.1145/3034786.3056125

R

A	B	C	*tid*
a	b	c	p
d	b	c	r

$\pi_{\text{B}}(\pi_{\text{AC}}R \bowtie \pi_{\text{BC}}R)$

B	*which*	*why*	*bag-why*
b	$\{p, r\}$	$\{\{p\}, \{p, r\}, \{r\}\}$	$\{[p, p], [p, r], [r, r]\}$

Figure 1: Which, why, or bag-why provenance

also use an important perspective cemented by [14]: provenance as *data annotation*. The query shown there has three distinct witnesses: $\{p\}, \{p, r\}$ and $\{r\}$. Each of these witnesses can lead to different reasoning about trust, cost, and probabilistic events.

The kind of provenance that works properly for probabilistic reasoning is, not surprisingly, given by boolean expressions (see, e.g., [44]). Motivated by conjunctive query equivalence, rather than by probabilistic reasoning, [14] also developed *minimal witness* provenance. For example, in the why-provenance $\{\{p\}, \{p, r\}, \{r\}\}$ the minimal witnesses are $\{p\}$ and $\{r\}$: either of them "absorbs" the witness $\{p, r\}$. Interestingly, it was realized later (see [30]) that minimal witness provenance is isomorphic to positive boolean expression provenance.

And yet, neither why-provenance nor boolean provenance is appropriate for A1 or A2 (trust or cost). The reason for this is that an input tuple might be used more than once in the derivation of an output. If so, the cost of that tuple should be doubled (and as we shall see, the trust/confidence can be squared). Therefore we need to track how many times each input item is used. With the hindsight afforded by the development of the semiring framework, an appropriate provenance here could be a variant of why- in which the witnesses are *bags* rather than sets. In Figure 1 we gave the local ad-hoc name *bag-why-provenance* to this approach. It collects $\{[p, p], [p, r], [r, r]\}$ where, for example, $[p, p]$ is the bag with two copies of p. [2] If, say, the tuple p costs 1$ and the tuple r costs 2$ then I wish to only pay 2$ to run this query since I can get the answer using just two copies of p as the bag-witness $[p, p]$ attests.

The approaches discussed so far are appropriate for set semantics but they do not capture bag semantics, specifically, not the multiplicity of output tuples. The semiring framework does.

Joint and alternative use of data.

In [34] a careful analysis of information flow through the operators of the positive relational algebra is performed. The result identifies two fundamental ways in which tuples are used: *jointly* as in cartesian product, join, selection, and intersection, and *alternatively* as in projection and union.

[2]In fact, these sets of bags are isomorphic to the polynomials with boolean coefficients introduced in [30]. There is a corresponding minimal-witness version for bag witnesses, the absorptive polynomials introduced in [24].

This leads naturally to consider a space of provenance annotations instrumented by two *abstract* binary operations: $+$ for alternative use, and $\cdot$ for joint use. The provenance space will also contain the annotations of the input tuples. More flexibility is gained if we do not think of them as tuple ids. We use the term *provenance tokens* and allow multiple tuples (e.g., all the tuples in the same table) to be annotated with the same token if useful.

For the query in Figure 1 the annotation of the output, so far, is $p \cdot (p + r) + r \cdot (p + r)$.

We think of $\sigma_P R$ as intersecting R with a virtual relation W_R that consists of all the possible tuples over the active domain of R. We postulate two constant provenance annotations, 0 and 1, and we annotate those tuples in W_R that satisfy P with 1 and those which do not with 0. So, for example, for the relation R in Figure 1 and the query $\pi_B \sigma_{\text{A}=d}(\pi_{\text{AC}}R \bowtie \pi_{\text{BC}}R)$ the annotation of the output, so far, is $(p \cdot (p + r)) \cdot 0 + (r \cdot (p + r)) \cdot 1$.

As shown in [34] the algebraic identities that we may desire for the algebra of annotated relations determine a specific structure on the space K of annotations: $(K, +, \cdot, 0, 1)$ is a **commutative semiring**.

Now we can use the semiring laws to revisit the examples given before. For the query in Figure 1 the annotation of the output can be written as a multivariate polynomial in indeterminates p and r: $p^2 + 2pr + r^2$ while for the query $\pi_B \sigma_{\text{A}=d}(\pi_{\text{AC}}R \bowtie \pi_{\text{BC}}R)$ we have $pr + r^2$. These *provenance polynomials* are called *how-provenance* in [16]. They can also be seen as bags of bag-witnesses.

Commutative semirings.

$(K, +, \cdot, 0, 1)$ with $0 \neq 1$, is a *semiring* when $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, $\cdot$ distributes over $+$ and $0 \cdot a = a \cdot 0 = 0$. The semiring is *commutative* when $\cdot$ is commutative. The semiring is *idempotent* when $+$ is idempotent.

Any distributive lattice is an idempotent commutative semiring. Here are some commutative semirings of interest to us:

1. $\mathbb{B} = (\mathbb{B}, \vee, \wedge, \bot, \top)$ is the standard habitat of logical truth. It is a distributive lattice and it corresponds to *set semantics* in databases.
2. $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ is used for *bag semantics* and can also be used for *counting derivations*.
3. $\mathbb{T} = (\mathbb{R}^{\infty}_{+}, \min, +, \infty, 0)$, is called the *tropical* semiring and is idempotent but not a distributive lattice. Its elements and operations appear in *min-cost* interpretations (e.g., shortest paths) and it plays a surprising role in connecting certain dynamic programming algorithms in statistics with certain methods of algebraic geometry [43] (see also next item).
4. $\mathbb{V} = ([0, 1], \max, \cdot, 0, 1)$ is called the *Viterbi* semiring and is isomorhic to $\mathbb{T}$ via $x \mapsto e^{-x}$ and $y \mapsto -\ln y$. When interpreted as probabilities, its elements and operations appear in *statistical model* interpretations (e.g., maximum probability trajectories in Hidden Markov Models). We can think of the elements of $\mathbb{V}$ as *confidence scores*, and use them in a quantitative approach to trust.
5. $\mathbb{F} = ([0, 1], \max, \min, 0, 1)$, is called the *fuzzy* semiring. It is a distributive lattice.

6. $\mathbb{A} = (\{\mathsf{P} < \mathsf{C} < \mathsf{S} < \mathsf{T} < 0\}, \min, \max, 0, \mathsf{P})$ is the *access control* semiring
   P is "public"
   C is "confidential"
   S is "secret"
   T is "top secret"
   0 is "so secret that nobody can access it!"
   This is a distributive lattice (beware! the lattice order is the opposite of the one we used in the definition). This is a just an illustrative example of clearance levels. Any finite distributive lattice would do. Interestingly, non-distributive lattices may lead to wrong permissions depending on optimizations performed by the DBMS.

7. $\mathbb{N}[X] = (\mathbb{N}[X], +, \cdot, 0, 1)$, denoted $\mathbb{N}[X]$, consist of multivariate polynomials in indeterminates from X and with coefficients from $\mathbb{N}$. This is the commutative semiring freely generated by the set X. It's used for a general form of provenance: the provenance polynomials.

8. $(\mathsf{PosBool}(X), \vee, \wedge, \bot, \top)$, denoted $\mathsf{PosBool}(X)$, whose elements are classes of equivalent positive (monotone) boolean expressions with boolean variables from X (its elements are in bijection with the positive boolean expressions in irredundant disjunctive normal form). This is the distributive lattice freely generated by the set X. Also used for provenance, e.g., in probabilistic databases.

K-sets and linear (semi)algebra.

In what follows we fix a commutative semiring $(K, +, \cdot, 0, 1)$. For a set D and a function $A : D \to K$ we define the *support* of A

$$\mathsf{supp}(A) \;=\; \{x \in D \mid A(x) \neq 0\}$$

We now define a **K-set** with elements from D to be a function $D \to K$ that has *finite support*. The straightforward definition of addition of K-sets

$$(A + B)(x) \;=\; A(x) + B(x)$$

determines a commutative monoid structure on K-sets with $\mathsf{0}(x) = 0$. Notice that $\mathbb{B}$-sets are ordinary sets whose addition is union while $\mathbb{N}$-sets are bags with bag union. For a finite arity r we can define [34] a *K-relation* to be a $\mathbb{D}^r$-set where $\mathbb{D}$ is a database domain.

In addition to forming a commutative monoid, the K-sets also can be "multiplied" by elements of K, an operation similar to multiplication by scalars in a vector space. For any $k \in K$ and any K-set A we define:

$$(k * A)(x) \quad = \quad k \cdot A(d)$$

This puts a **K-semimodule** structure (see [8, 47]) on K-sets. In the semimodule of K-sets the singletons form a *basis*. Indeed, for each $d \in D$ define $\{d\} : D \to K$

$$\{d\}(x) \;=\; \mathit{if}\; x = d \;\mathit{then}\; 1 \;\mathit{else}\; 0$$

Then, any K-set A such that $\mathsf{supp}(A) = \{d_1, \ldots, d_n\}$ can be written as a linear combination of singletons

$$A \;=\; A(d_1) * \{d_1\} + \cdots + A(d_n) * \{d_n\}$$

Now the definition of cross-product of K-relations (similar to [34]) is forced by a postulate that requires $\times$ to be *bilinear*, together with

$$(k * \{r\}) \times (l * \{s\}) = (k \cdot l) * \{\langle r, s\rangle\}$$

which follows from $k \cdot l$ corresponding to joint use.

With 20/20 hindsight, regarding K-sets as linear combinations also explains the extension of the semiring framework from flat to nested relational query languages [27]. It can also partially explain the extension to aggregation [8]. There, aggregation domains are commutative monoids with operations such as min, max or summation. Regarding these monoids as semimodules is an essential insight. The other important component of the aggregation work is the *tensor product* construction $K \otimes M$ which extends any commutative monoid M to a K-semimodule. Similar to the K-sets, the elements of $K \otimes M$ can be expressed as a linear combination

$$k_1 * \iota(m_1) + \cdots + k_n * \iota(m_n)$$

where $\iota : M \to K \otimes M$ maps the elements of M to tensors that generate the entire space (M represents a basis when ι is injective). This gives us a general idea of *K-annotated aggregations* that also covers "sums" of K-sets as a particular case. See [47] for more details and note also the interesting connections with [38].

Semiring homomorphisms and query equivalence.

K-relations give a whole family of standard ($K = \mathbb{B}$ and $K = \mathbb{N}$) and nonstandard (all other K of interest) semantics for relational query languages and beyond. For each K, such a **K-semantics** yields a corresponding notion of *query equivalence*. [3] Pleasingly (even sanely), K-semantics for many query languages satisfies the following **fundamental property**. We state it for the positive relational queries [34] but it also holds for Datalog [34], for the Nested Relational Calculus and its extension to trees [27], for unordered XQuery [27], for extension to aggregates [8], and for FOL model checking [48].

Given two commutative semirings and a homomorphism between them $h : K_1 \to K_2$ we lift h to map K_1-relations, and even K_1-databases, to K_2-relations, respectively K_2-databases: simply by replacing an annotation $k \in K_1$ with $h(k) \in K_2$. Then, for any positive relational query (SPJRU or UCQ), Q the following diagram commutes:

$$\begin{array}{ccc} K_1\text{-db} & \xrightarrow{\;h\;} & K_2\text{-db} \\ \Big\downarrow Q & & \Big\downarrow Q \\ K_1\text{-rel} & \xrightarrow{\;h\;} & K_2\text{-rel} \end{array}$$

Equivalence of positive relational queries (in the form of UCQs—unions of conjunctive queries) for various K was studied in [30, 31].

Of particular interest are the cases when K is used for capturing provenance. We have already mentioned two such semirings: $\mathbb{N}[X]$, the semiring of provenance polynomials,

[3] And, it turns out, also of *query containment*. However, the latter relies on *naturally ordered* semirings, a topic we shall not discuss here.

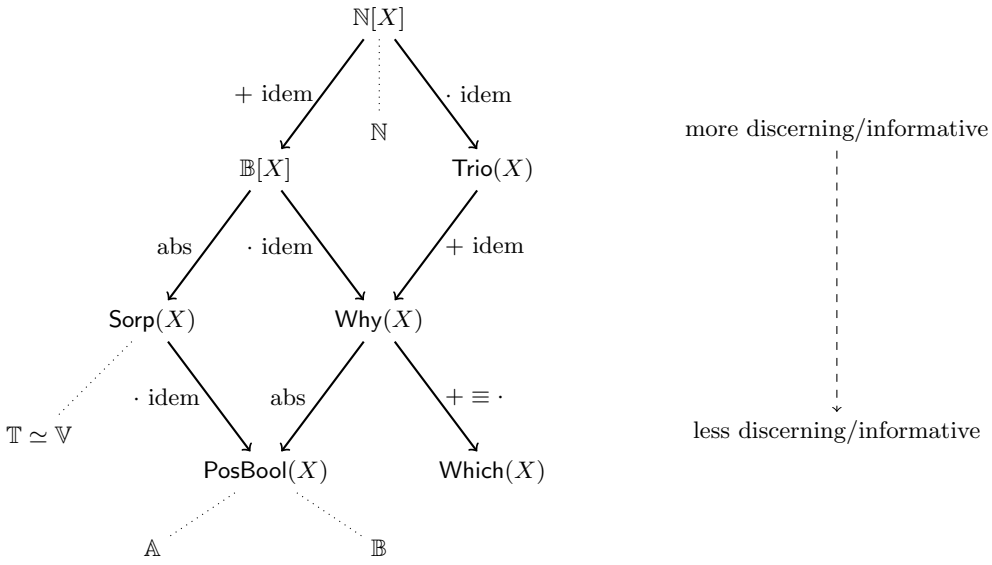

Figure 2: A hierarchy of provenance semirings

and ($\mathsf{PosBool}(X)$, the semiring of positive boolean expressions (modulo equivalence). The former captures the most general form of provenance available in the semiring framework while the latter captures provenance that suffices for the applications A3 and A4 and for certain aspects of the applications A5 and A6 (A1-A6 are mentioned at the beginning of this paper). A couple of other commutative semrings have been deemed useful for expressing provenance. For example, it is shown in [30, 31] how to capture the which-provenance, the why-provenance, and the bag-why-provenance (recall Figure 1) in specific commutative semirings (called there $\mathsf{Lin}(X), \mathsf{Why}(X)$ and respectively $\mathbb{B}[X]$, the latter consisting of polynomials similar to those in $\mathbb{N}[X]$, except with boolean coefficients). An appropriate semiring, $\mathsf{Trio}(X)$, is also described for the provenance defined in [9]. [4] Another semiring, $\mathsf{Sorp}(X)$ consists of the *absorptive* polynomials and is the absorptive commutative semiring freely generated by X (the absorption law is $a + a \cdot b = a$). It was introduced in [24] and while it does not support bag semantics, it is nicely appropriate for applications A1-A6 above, consisting of minimal bag-witnesses.

We shall say that two relational queries are **K-equivalent** if they give the same result when applied to databases consisting of K-relations. It is a consequence of the fundamental property shown above that when we have a *surjective* homomorphism $h : K_1 \to K_2$ then K_1-equivalence of UCQs implies their K_2-equivalence. Thus, surjective homomorphisms offer a tool for comparing semirings to understand (to some extent) how *discerning* the various K-semantics are. Their presence means "at least as discerning" (but not necessarily "strictly more discerning"), see [30, 31]. We show this in Figure where the solid arrows indicate the presence of a surjective homomorphism. Moreover, these homomorphisms preserves the provenance tokens and all the digrams shown commute. The arrows are labeled with the algebraic laws whose addition characterizes the passage from a more discerning to a less discerning provenance semiring. For example, making both + and · idempotent gives us $\mathsf{Why}(X)$. This semiring can be shown to be the commutative, +-idempotent, and ·-idempotent semiring generated by X. Further adding absorption gives the free distributive lattice $\mathsf{PosBool}(X)$. As expected less discerning corresponds to less "informative" provenance. Here is an example polynomial mapped along two paths of DAG of surjective homomorphisms (the mappings not shown can be deduced):

$$\begin{aligned} 2p^2r + pr + 5r^2 + ps \in \mathbb{N}[X] &\mapsto p^2r + pr + r^2 + ps \in \mathbb{B}[X] \\ &\mapsto pr + r^2 + ps \in \mathsf{Sorp}(X) \\ &\mapsto r + ps \in \mathsf{PosBool}(X) \end{aligned}$$

and

$$\begin{aligned} 2p^2r + pr + 5r^2 + ps \in \mathbb{N}[X] &\mapsto 3pr + 5r + ps \in \mathsf{Trio}(X) \\ &\mapsto pr + r + ps \in \mathsf{Why}(X) \\ &\mapsto prs \in \mathsf{Which}(X) \end{aligned}$$

For more details, a study of query containment and complexity results see [30, 31].

Finally, the dotted arrows in Figure connect the "application" semirings, $\mathbb{B}, \mathbb{A}, \mathbb{T}, \mathbb{V}$ and $\mathbb{N}$ with the least informative provenance semiring that can still be used for their application. For example, any assignment of values in, say, $\mathbb{V}$ to provenance tokens in X extends uniquely to an (evaluation) homomorphism from $\mathsf{Sorp}(X)$ to $\mathbb{V}$.

Some relevant papers.

Here are some useful surveys of provenance and its impact in several domains: [21, 16, 17, 3].

The original motivation for the semiring framework came from the need to manage trust in data sharing as formulated by Z. Ives et al. [36]. The first paper [34] dealt with the positive relational algebra and Datalog. It was followed by investigations into provenance for nested relations/complex values (objects) as well as XQuery (for unordered XML) [27],

[4] We should warn the reader against a possible confusion with the terminology in this field. Which-provenance is called *lineage* in [18, 19], but lineage was used in [9] for a different kind of provenance, and yet again in [44] for what we called here $\mathsf{PosBool}(X)$-provenance.

SQL aggregates [8], workflows with map-reduce modules [4], and languages for data-centric (data-dependent) processes [25], SemanticWeb's SPARQL [29], and even matrix operations [51].

The semiring approach has been successfully implemented in two software systems, Orchestra [33, 35] and Propolis [25]. In addition, access control in Facebook's search feature uses a provenance-based scheme very similar in spirit to the semiring approach [20]. Importantly, space-efficient implementations of semiring provenance use the graph representations nicely described in [26]. The design and implementation of a language for querying provenance graphs was presented in [37]. The size of provenance information has been an early and continuous concern for the semiring framework, both theoretically [5, 6, 24], and practically [2, 1].

The core theoretical underpinnings of the semiring framework (query containment and equivalence properties, constructions of semirings, fixed points, semimodules, and tensor products) have been developed mostly in [34, 27, 30, 8, 32, 47, 25]. The reader may have noticed that the bulk of this work was concerned with *positive* query languages. Indeed, trying to add to the commutative semiring structure operations that capture negation or difference of relations has led to interesting but divergent approaches [28, 32, 8, 7, 29] and a somewhat puzzling state of affairs. Recent work done by E. Grädel and the second author [48] deals with negation syntactically, via *negation normal forms* and thus supports full duality, and, for example, tuple insertion, and why-not provenance.

We note the interesting line of work on *where-provenance* which tracks the input items that get copied as part of the output [14, 13]. Tuple-level annotations will not, in general, be able to track the copying of individual field values. However, a more flexible annotation scheme combined with semiring provenance can [46].

The use of Datalog extensions to specify network configuration rules [40] has already led to many applications of provenance in this area, e.g. [54, 53, 50, 49, 15]. Similarly, the use of Datalog in formal verification [52, 41] is likely to benefit from provenance analysis [42].

The semiring framework has also played a role in motivating recent work on "what-if analysis" [23, 12], "query-by-explanation" [22], and "missing data" [45].

There exists a well-known tight connection between conjunctive queries in databases and constraint satisfaction problems in AI [39]. In this light, and despite a number of technical differences, there exists an interesting connection (that needs more exploration) between the semiring provenance framework applied to conjunctive queries and the semiring framework for *soft constraint satisfaction* [11, 10].

Acknowledgements.

None of this would have been possible without the crucial contributions of our closest collaborators (so far, and in chronological/alphabetical order): Zack Ives, Grigoris Karvounarakis, Nate Foster, Yael Amsterdamer, Daniel Deutch, Tova Milo, Sudeepa Roy, Yuval Moskovitch, Allen Yan, and Erich Grädel. We are also very grateful to others with whom we have collaborated on papers with related topics: Susan Davidson, Julia Stoyanovich, Sanjeev Khanna, Shan Shan Huang, Boon Thau Loo, Molham Aref, Vittorio Perduca, Sepehr Assadi, Yang Li, Jeff Naughton, Bruhathi Sundarmurthy, and Paris Koutris. Val Tannen was partially supported by NSF 1302212 and 1547360, by NIH U01EB020954-01 as well as by the Simons Institute for the Theory of Computing.

1. REFERENCES

[1] E. Ainy, P. Bourhis, S. B. Davidson, D. Deutch, and T. Milo. Approximated summarization of data provenance. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 483–492, 2015.

[2] E. Ainy, S. B. Davidson, D. Deutch, and T. Milo. Approximated provenance for complex applications. In *6th Workshop on the Theory and Practice of Provenance, TaPP'14, Cologne, Germany, June 12-13, 2014*, 2014.

[3] I. Altintas, M. K. Anand, D. Crawl, S. Bowers, A. Belloum, P. Missier, B. Ludäscher, C. A. Goble, and P. M. A. Sloot. Understanding collaborative studies through interoperable workflow provenance. In *Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers*, pages 42–58, 2010.

[4] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. Putting lipstick on pig: Enabling database-style workflow provenance. *PVLDB*, 5(4):346–357, 2011.

[5] Y. Amsterdamer, D. Deutch, T. Milo, and V. Tannen. On provenance minimization. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 141–152, 2011.

[6] Y. Amsterdamer, D. Deutch, T. Milo, and V. Tannen. On provenance minimization. *ACM Trans. Database Syst.*, 37(4):30, 2012.

[7] Y. Amsterdamer, D. Deutch, and V. Tannen. On the limitations of provenance for queries with difference. In *3rd Workshop on the Theory and Practice of Provenance, TaPP'11, Heraklion, Crete, Greece, June 20-21, 2011*, 2011. See also CoRR abs/1105.2255.

[8] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 153–164, 2011. See also CoRR abs/1101.1110.

[9] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[10] S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume 2962 of *Lecture Notes in Computer Science*. Springer, 2004.

[11] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.

[12] P. Bourhis, D. Deutch, and Y. Moskovitch. Analyzing data-centric applications: Why, what-if, and how-to. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 779–790, 2016.

[13] P. Buneman, J. Cheney, and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. In *Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings*, pages 209–223, 2007.

[14] P. Buneman, S. Khanna, and W. C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[15] A. Chen, Y. Wu, A. Haeberlen, B. T. Loo, and W. Zhou. Data provenance at internet scale: Architecture, experiences, and the road ahead. In *Conference on Innovative Data Systems Research (CIDR'17)*, Jan. 2017.

[16] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4), 2009.

[17] J. Cheney, S. Chong, N. Foster, M. I. Seltzer, and S. Vansummeren. Provenance: a future history. In *Companion to the 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2009, October 25-29, 2009, Orlando, Florida, USA*, pages 957–964, 2009.

[18] Y. Cui and J. Widom. Practical lineage tracing in data warehouses. In *ICDE*, pages 367–378, 2000.

[19] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, 2000.

[20] M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. B. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss, C. Yang, and N. Zhang. Unicorn: A system for searching the social graph. *PVLDB*, 6(11):1150–1161, 2013.

[21] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.

[22] D. Deutch and A. Gilad. Qplain: Query by explanation. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1358–1361, 2016.

[23] D. Deutch, Z. G. Ives, T. Milo, and V. Tannen. Caravan: Provisioning for what-if analysis. In *CIDR*, 2013.

[24] D. Deutch, T. Milo, S. Roy, and V. Tannen. Circuits for datalog provenance. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014.*, pages 201–212, 2014.

[25] D. Deutch, Y. Moskovitch, and V. Tannen. Provenance-based analysis of data-centric processes. *VLDB J.*, 24(4):583–607, 2015.

[26] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[27] J. N. Foster, T. J. Green, and V. Tannen. Annotated XML: queries and provenance. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 271–280, 2008.

[28] F. Geerts and A. Poggi. On database query languages for k-relations. *J. Applied Logic*, 8(2):173–185, 2010.

[29] F. Geerts, T. Unger, G. Karvounarakis, I. Fundulaki, and V. Christophides. Algebraic structures for capturing the provenance of SPARQL queries. *J. ACM*, 63(1):7:1–7:63, 2016.

[30] T. J. Green. Containment of conjunctive queries on annotated relations. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 296–309, 2009.

[31] T. J. Green. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, 49(2):429–459, 2011.

[32] T. J. Green, Z. G. Ives, and V. Tannen. Reconcilable differences. *Theory Comput. Syst.*, 49(2):460–488, 2011.

[33] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 675–686, 2007.

[34] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40, 2007.

[35] Z. G. Ives, T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. C. N. Pereira. The ORCHESTRA collaborative data sharing system. *SIGMOD Record*, 37(3):26–32, 2008.

[36] Z. G. Ives, N. Khandelwal, A. Kapur, and M. Cakir. ORCHESTRA: rapid, collaborative sharing of dynamic data. In *CIDR 2005, Second Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings*, pages 107–118, 2005.

[37] G. Karvounarakis, Z. G. Ives, and V. Tannen. Querying data provenance. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 951–962, 2010.

[38] C. Koch. Incremental query evaluation in a ring of databases. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 87–98, 2010.

[39] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.

[40] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking. *Comm. ACM*, 52(11):87–95, 2009.

[41] R. Mangal, X. Zhang, A. V. Nori, and M. Naik. A user-guided approach to program analysis. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pages 462–473, 2015.

[42] M. Naik. Private communication.

[43] L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press,

2005.

[44] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[45] B. Sundarmurthy, P. Koutris, W. Lang, J. Naughton, and V. Tannen. m-tables: Representing missing data. In *ICDT*, 2017. To appear.

[46] V. Tannen. Provenance for database transformation. 2010 EDBT/ICDT Keynote slides `www.cis.upenn.edu/~val/EDBTkeynoteLausanne.pdf`.

[47] V. Tannen. Provenance propagation in complex queries. In *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*, pages 483–493, 2013.

[48] V. Tannen. Provenance analysis for FOL model checking (Semantics Column, Mike Mislove, ed.). *ACM SIGLOG News*, 4(1):24–36, 2017.

[49] Y. Wu, A. Chen, A. Haeberlen, W. Zhou, and B. T. Loo. Automated network repair with meta provenance. In *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*, Mar. 2017.

[50] Y. Wu, M. Zhao, A. Haeberlen, W. Zhou, and B. T. Loo. Diagnosing missing events in distributed systems negative provenance. In *Proceedings of ACM SIGCOMM 2014*, Aug. 2014.

[51] Z. Yan, V. Tannen, and Z. G. Ives. Fine-grained provenance for linear algebra operators. In *8th USENIX Workshop on the Theory and Practice of Provenance (TaPP 16)*, 2016.

[52] X. Zhang, R. Mangal, R. Grigore, M. Naik, and H. Yang. On abstraction refinement for program analyses in datalog. In *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '14, Edinburgh, United Kingdom - June 09 - 11, 2014*, pages 239–248, 2014.

[53] W. Zhou, L. Ding, A. Haeberlen, Z. Ives, and B. T. Loo. TAP: Time-aware provenance for distributed systems. In *Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11)*, June 2011.

[54] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr. Secure Network Provenance. In *Proc. SOSP*, 2011.