A Brief Tour through Provenance in Scientific Workflows and Databases

Bertram Ludäscher

Graduate School of Library and Information Science
& National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
ludaesch@illinois.edu

Abstract

Within computer science, the term *provenance* has multiple meanings, due to different motivations, perspectives, and assumptions prevalent in the respective communities. This chapter provides a high-level "sightseeing tour" of some of those different notions and uses of provenance in scientific workflows and databases.

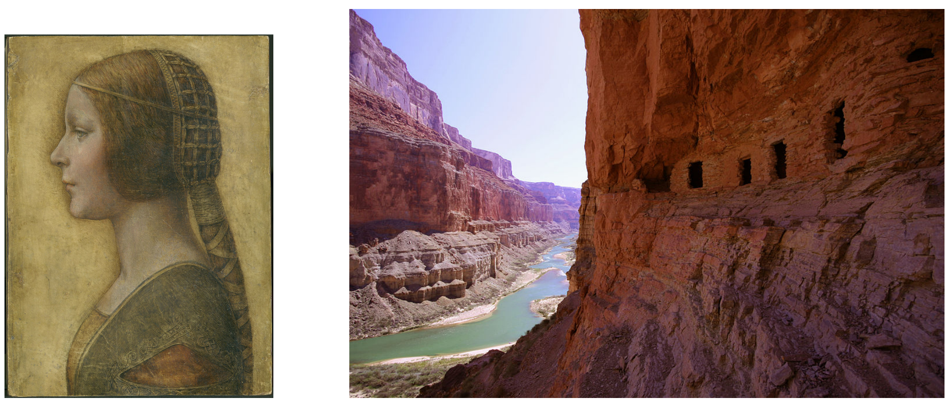
Keywords: prospective provenance, retrospective provenance, lineage, provenance polynomials, why-not provenance, provenance games.

Contents

1 Provenance in Art, Science, and Computation

The Oxford English Dictionary (OED) defines provenance as "*the place of origin or earliest known history of something; the beginning of something's existence; something's origin*." Another meaning listed in the OED is "*a record of ownership of a work of art or an antique, used as a guide to authenticity or quality*."

In the fine arts, the importance of this notion of provenance can often be measured with hard cash. For example, one of Picasso's *Les Femmes d'Alger* sold for nearly $180 million in May 2015 at Christie's in New York; a new record for a painting at an auction. In contrast, *La Bella Principessa* sold for less than $20,000 in 2007, despite the fact that some attribute it to the great Leonardo da Vinci (Figure 1(a)). However, there is no documented *chain of custody* prior to the 20th century, so the drawing's incomplete provenance record is insufficient to establish its authenticity. It is now up to "provenance sleuths" to try and determine whether or not the drawing was really created by da Vinci – in which case it could rival the value of *Les Femmes d'Alger*.



(a) (b)

Figure 1: Provenance in the Arts and Sciences: (a) *La Bella Principessa*, portrait by Leonardo da Vinci. Or is it? It could be worth well over $100 million dollars, *if* enough provenance were available to verify its authenticity. (b) Grand Canyon's rock layers are a record of the early geologic history of North America. The ancestral puebloan granaries at Nankoweap Creek tell archaeologists about the much more recent human history. (By Drenaline, licensed under CC BY-SA 3.0)

Scientists often have to be expert provenance sleuths themselves. As part of conducting their science they may, e.g., analyse the stratigraphy of the Grand Canyon in order to reveal the geologic history of the planet (Figure 1(b)), or study the fossil record preserved in rock layers or the molecular record inscribed in the DNA of species to reconstruct phylogenies and assemble the tree of life. Empirical evidence plays a crucial role in the scientific method and is a form of provenance that is everywhere around us, from the cosmic microwave background left behind by the Big Bang, to the recurrent laryngeal nerve we share with all tetrapods [Wed11] – clear evidence of our common lineage with all life [Dob73].

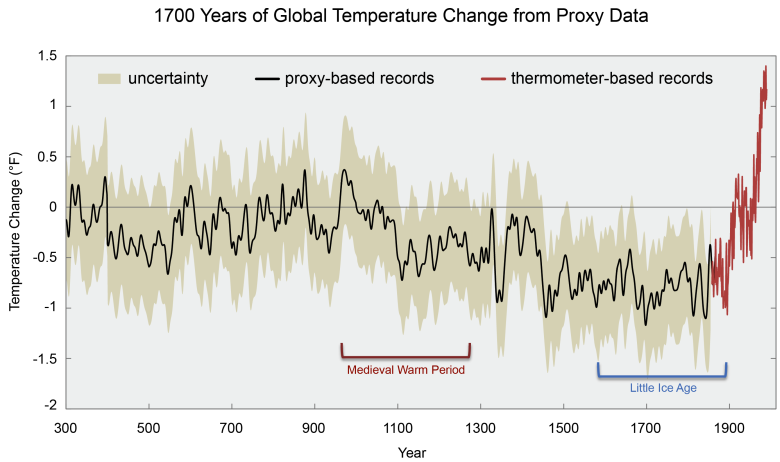


Figure 2: "Hockey stick" graph from [MRY14] (adapted in turn from [MZH+08]) showing temperature changes of the Northern Hemisphere from observations (red) and proxies (black) relative to the 1961–1990 average temperature (gray 0° F line).

1.1 Transparency and Reproducibility in Science

It is long standing practice to cite your sources in scientific publications. However, as science has become increasingly computational and data-driven [HTT09], and more interdisciplinary and collaborative, new requirements and opportunities have emerged for research articles. The U.S. Global Change Research Program (USGCRP) has developed the Global Change Information System (GCIS) [GCI15] that links global change information across many federal agencies. An important product of USGCRP is the National Climate Assessment (NCA) report [MRY14] which summarizes impacts of climate change on the U.S., now and in the future. To facilitate transparency and usability of the NCA, ambitious transparency goals have been set, ranging from basic source traceability (references to papers) to the use of data citations and metadata, all the way to traceable processes and software tools, with the ultimate goal to support full reproducibility of all NCA content [TFM+13].

Data provenance, the lineage and processing history of data, is of critical importance for transparency, to assess data quality [Sad13], and for computational reproducibility. Consider, e.g., the famous "hockey stick" graph in Figure 2, showing temperature changes over the last 1700 years. Similar to *La Bella Principessa*, the value of such a chart may depend on its provenance, in particular, on the quality of the data that went into it, and the soundness of the computational method used to create the final result. As scientists provide detailed provenance information, e.g., *what* proxy records where used to reconstruct past temperature data and *how* those proxies where processed to derive a temperature, other scientists can evaluate and assess the results and the validity of the findings.

In a recent article, Hill *et al.* [HDD+15] make a strong case for data provenance for science. They cite a study by Eisenman *et al.* [EMN14] that argues that the Antarctic sea ice extent was probably not growing nearly as fast as thought, and that *"much of this [ice] expansion may be a spurious artifact of an error in the processing of the satellite obser-*

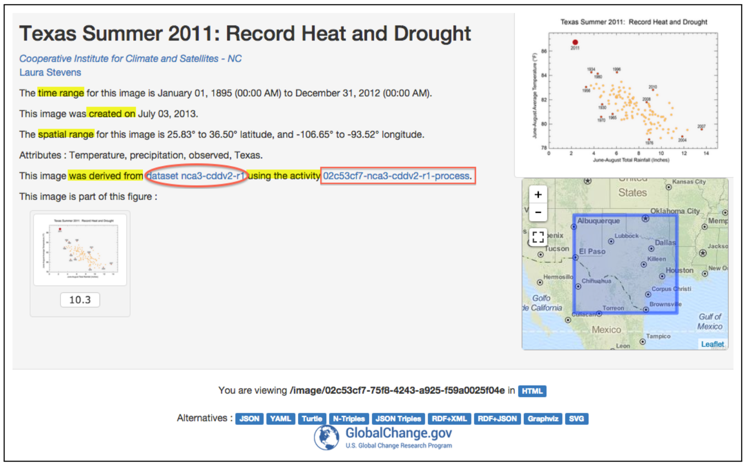


Figure 3: Screenshot from data.globalchange.gov, showing a rainfall *vs* temperature scatter plot for Texas between 1895 and 2012 (upper right); provenance metadata (center) with links to the source data (highlighted oval) and software (highlighted rectangle) used to create the plot [Ste13].

vations." Hill *et al.* also report that ESIP[1] seeks to accelerate the implementation of new approaches to track all details necessary to demonstrate data validity and to ensure scientific reproducibility using a Provenance and Context Content Standard (PCCS) [HDD+15].

The third NCA report provides some of the much needed provenance and context information through the related GCIS system. Figure 3 depicts a screenshot showing rainfall *vs* temperature data. Metadata provided for the scatter plot in the upper right of the figure includes its spatial extent (lower right) and its temporal extent (the years from 1895 to 2012). Last not least, provenance links to the original dataset and software are highlighted in this HTML metadata view as well. By pushing one of the buttons at the bottom of the screen, this metadata can also be exposed in one of several other machine-readable formats, including JSON, YAML, Turtle, and RDF. While this rich metadata and provenance information is clearly useful and required for transparency, the compilation of this information for the report and the GCIS system required an extraordinary three-year effort by a team of more than 300 experts [MRY14]. As more and more workflow tools and scripting environments become "provenance-enabled", the capture, sharing, and querying of provenance information in support of reproducible science should become easier as well.

2 Provenance in Scientific Workflows

A scientific workflow is a description of a process for accomplishing a scientific objective, usually expressed in terms of tasks and their dependencies [LBM09]. Such workflows aim to support computational science and accelerate scientific discovery in various ways, e.g., by providing process ***A****utomation*, ***S****calable execution*, ***A****bstraction*, and ***P****rovenance* support (ASAP for short) [CVDK+12]. The latter, i.e., the automated tracking of prove-

[1] The Federation of Earth Science Information Partners

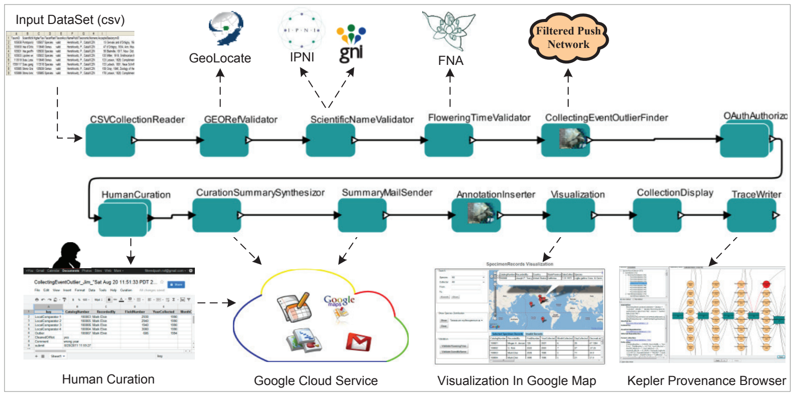


Figure 4: Kepler data curation workflow for specimen data [DCM+12]. The workflow graph itself represents *prospective* provenance. The trace graph (*retrospective* provenance) depicted in the lower right can be viewed with a separate application; see Figure 5.

nance is often considered one of the key advantages of using a workflow system for process automation [DBE+07, Bow12].

Common processing examples include data formatting, subsetting, cleaning, and analysis. Compute-intensive workflows often result from computational science *simulations*, e.g., running climate and ocean models, or other simulations from particle-physics, chemistry, biology, to ecology, astronomy, and cosmology [LAB+09]. Scientific workflows can be simple, linear chains of tasks, but more complex dataflow graphs are also common [MBZL09].

2.1 Workflows as Prospective Provenance

Figure 4 depicts an example scientific workflow for the semi-automatic curation of specimen collections data [DCM+12], implemented using the Kepler scientific workflow system [LAB+06]. In Kepler, computational steps execute independently from one another and are implemented by so-called (software) *actors* (green boxes in Fig. 4). These actors are connected via dataflow *channels* that are typically implemented using FIFO (first-in first-out) buffers, i.e., in such workflows data elements can be executed in pipeline-parallel mode, similar to the way a UNIX pipeline executes. The workflow in Figure 4 reads as input a CSV file containing specimen records from a natural history collection. Such biodiversity datasets may require time-consuming, manual data curation steps. Using workflow tools, a number of data quality control measures and repair suggestions can be processed more efficiently. Here, the workflow checks various fields of the data records as they are streamed through the process pipeline, e.g., the plausibility of geolocation information (where a specimen was collected), the scientific name of the specimen, and the flowering time (for plants an additional check on the collection date). Further downstream, human actors are involved in checking the records flagged by upstream computational steps [DCM+12]. The final steps of the workflow display record locations on a map and output a *provenance graph* that can be queried and explored in a separate provenance browser [BMR+08, ABL10a].

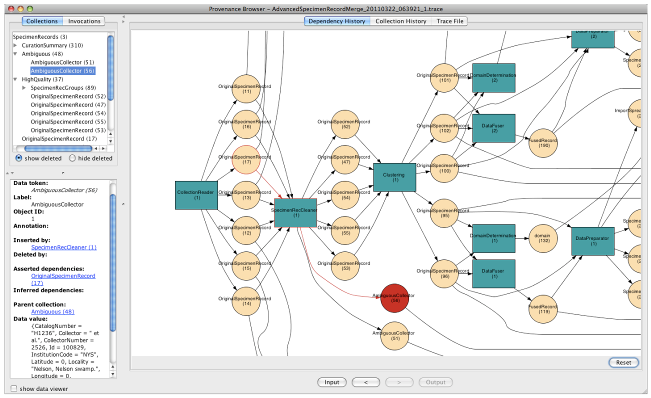


Figure 5: Kepler Provenance Browser [BMR+08, ABL10a]: A *retrospective provenance* graph (recorded earlier, during workflow execution) is displayed and can be navigated forward and backward in time via VCR-like control buttons (bottom).

The curation workflow graph depicted in Figure 4 provides an overall description of the processing steps that a data record will undergo when subjected to the workflow. In this way, workflows are a form of *prospective provenance*: the workflow graph captures the general method or "recipe" of how data products of a workflow are processed. When a computational method is documented in this way, as a workflow graph, users can already make certain inferences about the general method and about the result data produced by it. For example, from the graph in Figure 4 we see that the flowering time validation step (FNA) may use the improved geolocation data (GeoLocate) or a validated scientific name (IPNI/gni) since those upstream actors may have updated a record by the time it reaches the FNA step. Conversely, as the FNA actor lies downstream from GeoLocate and IPNI/gni, it *cannot* possibly influence the latter. Thus, while detailed dependency and lineage information between concrete data products is available only after workflow execution, some lineage information, in particular about the *independence* of steps can be obtained prior to execution, by querying the workflow graph. If a workflow graph contains further configuration information, e.g., which XML elements of a data stream are processed at each step, then a more detailed prospective provenance graph can be inferred as well [ZL10].

2.2 Retrospective Provenance from Workflow Execution Traces

Prospective provenance, in the form of a workflow graph, constitutes a first valuable knowledge artifact, documenting a computational method or workflow. Many workflow systems also allow users to record provenance information at runtime, i.e., they capture *retrospective provenance* that can be queried, analyzed, and visualized to gain a deeper understanding of how certain results were obtained as the workflow executed. Figure 5 depicts a screenshot of the Kepler Provenance Browser [BMR+08, ABL10a], showing retrospective provenance from a run of a specimen curation workflow similar to the one in Figure 4. Selected nodes and incident edges are highlighted to indicate which upstream step has generated a data item, and which downstream step(s) read it. Note that a single actor in

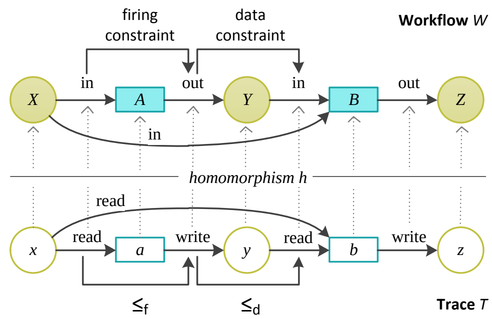


Figure 6: A homomorphism h from trace T to workflow W guarantees structural validity. Workflow-level constraints induce temporal constraints $\leq_\mathsf{f}$ and $\leq_\mathsf{d}$ on traces [DKBL12].

a prospective provenance graph can give rise to multiple *invocations* in the retrospective provenance graph, e.g., `DataFuser(1)` and `DataFuser(2)` in Figure 5 are two distinct invocations of a single `DataFuser` actor. Each invocation usually operates on its own data items (beige circles). Similarly, a single channel between connected actors in the workflow graph (prospective provenance) is often traversed by multiple data items which then appear as "data bundles" in the execution trace (retrospective provenance graph), as seen in Figure 5.

2.3 Models of Provenance and Scientific Workflows

In 2006 the scientific workflow community organized the first "Provenance Challenge" workshop to better understand the capabilities of different workflow systems and approaches [MLA+08]. The first workshop led to a number of follow-up challenge events (all set up to be informative rather than competitive), ultimately leading to the definition of the Open Provenance Model (OPM) [MFF+08, MCF+11], which in turn informed the development of the W3C PROV standard [MMB+12]. Much work in the scientific workflow community then focused on engineering challenges, e.g., the efficient storage [HA08, CJR08, ABML09], navigation [ABL09], and querying [ABL10b, ABL12] of provenance. When working with provenance in scientific workflows, the distinction between prospective and retrospective provenance is important. However, neither OPM nor its PROV successor deal with this distinction. One could argue that both OPM and PROV focus on retrospective provenance, but the underlying definitions are rather vague on that point.[2] As a result, different extensions to OPM and PROV have been developed that allow users to work with both prospective and retrospective provenance and relate both kinds of information in a single model [GG11, MDB+13].

2.4 Relating Retrospective and Prospective Provenance

It is often desirable to combine trace-level retrospective provenance and workflow-level prospective provenance in a single, uniform representation. Such a model should also

[2] For example, [MMB+12] states that *"provenance is defined as a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing."*

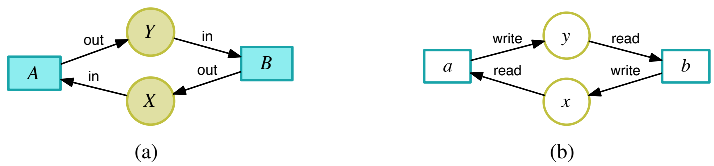


Figure 7: Cycles (a) in workflow W and (b) in trace T. A cycle (feedback loop) in a workflow is not uncommon, but in a trace it suggests a temporal inconsistency [DKBL12].

accommodate temporal information whenever available. This can be achieved with a semistructured data model, consisting of labeled, directed graphs of the form $G = (V, E, L)$, with vertices V, labels L, and labeled edges $E \subseteq V \times L \times V$. In the following, we view workflows (prospective provenance) W and traces (retrospective provenance) T as subgraphs of G. Similarly, a temporal model consists of labeled edges, modeling one or more "before" relations $\leq_R$.

Figure 6 shows a workflow W (top) and a trace T (bottom). By linking a trace to the workflow that generated it, important information can be obtained via the constraints of the combined model: If data item y is written into output container Y as a result of invocation a of actor A on input item x in X, then the writing of y cannot happen before x is read. Therefore, this *firing constraint* at the level of the workflow model W induces a corresponding *temporal constraint* on the trace T, i.e., $t_{\mathrm{read}(x)} \leq_{\mathsf{f}} t_{\mathrm{write}(y)}$. Similarly, the *data constraint* at the Y container in W induces another temporal constraint at the trace level: before item y can be read by invocation b of actor B, this item must first have been written by some invocation a of A, i.e., $t_{\mathrm{write}(y)} \leq_{\mathsf{d}} t_{\mathrm{read}(y)}$.

In [PMGF15] the authors use temporal information about the duration of interactions to exclude data dependencies that would violate temporal causality (if process A first writes y, then reads x, then y does not depend on x).

Structural and Temporal Constraints. The execution of workflow W in Figure 7(a) might have produced the trace T in Figure 7(b). To check whether T is indeed a possible instance of W, we link T's nodes and edges to W via a mapping h (as in Figure 6). For example, the edges $x \overset{\mathsf{read}}{\to} a$ and $a \overset{\mathsf{write}}{\to} y$ in T (x was read and y was written by invocation a), have correponding edges $X \overset{\mathsf{in}}{\to} A$ and $A \overset{\mathsf{out}}{\to} Y$ in the workflow W, linking data containers X and Y to the actor A. In Figure 7, T is structurally valid with respect to W, but other inconsistencies due to temporal constraints can still arise. For example, a cycle in T usually indicates an inconsistent trace: if read and write observables are temporally or causally linked, a *strict* partial order is implied and a cycle should not be observable. On the other hand, a cycle in W is usually *not* a concern. It simply means that W has a feedback loop, which is a rather common workflow pattern: loops in W are "unrolled" in T, leading to acyclic trace graphs T. In [DKBL12] we have formalized structural validity of a trace T via a homorphism $h : T \to W$ and shown that it can be checked using a simple Datalog query.[3] In [KMB15] a formal, temporal semantics of OPM is developed and it is shown that the original inference rules for OPM are sound but incomplete. In [DRL13] we have

[3]Here, we are not *searching* for a graph homorphism, but simply test whether the *given* mapping $h : T \to W$ is a homorphism.

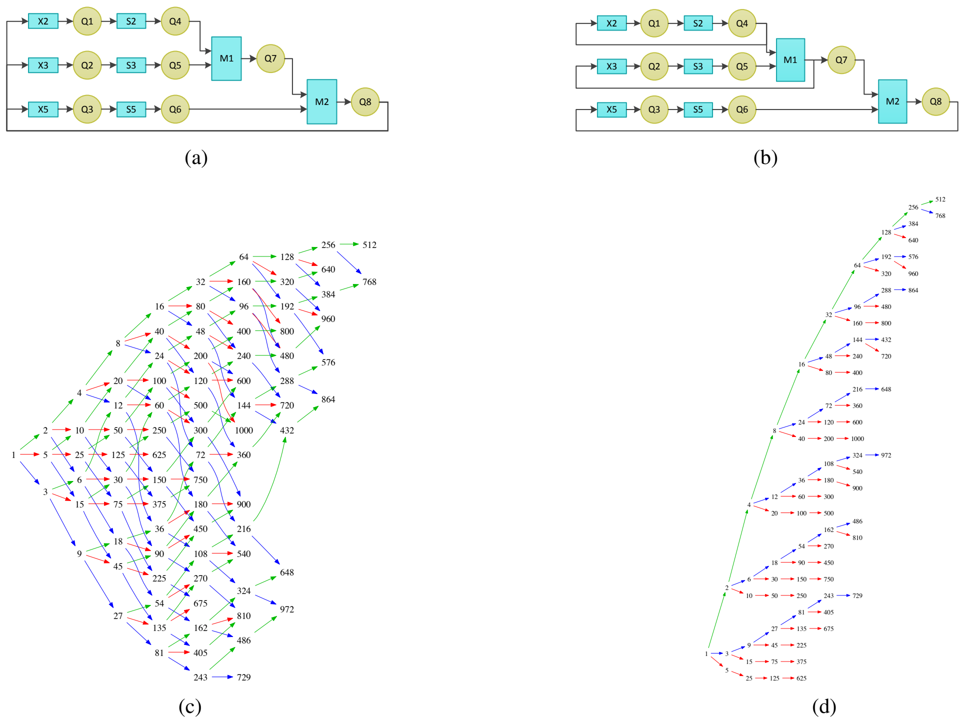


Figure 8: Hamming workflow variants (a) H_1 ("one loop") and (b) H_3 ("three loops"). Retrospective provenance can be used to spot inefficient, redundant workflow computations: (c) trace T_1 ("*Fish*") obtained by running H_1 and (d) trace T_3 ("*Sail*") from H_3. The many redundant lineage paths of the DAG in (c) match the regular path query $(\mathsf{x2} \mid \mathsf{x3} \mid \mathsf{x5})^*$, while the unique paths in the tree (d) satisfy the pattern $(\mathsf{x2}^* \cdot \mathsf{x3}^*) \cdot \mathsf{x5}^*$.

developed a rule-based implementation (inspired by [KMB15][4]) that allows provenance model engineers to experiment with different temporal semantics, expressed as constraints over the provenance model.

Example: Hamming Numbers. Consider the two variant workflows H_1 and H_3 in Figure 8(a) and 8(b) that compute *Hamming numbers*[5] [Dij81, Hem88]

$$H = \{2^i \cdot 3^j \cdot 5^k \mid i, j, k \geq 0\}$$

incrementally, i.e., as an ordered sequence $1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, \ldots$ While both workflows contain the same nodes (i.e., *actors* and *data containers*), they are wired slightly differently, which makes a big difference as it turns out. The data containers Q_i are queues (FIFO buffers); Q_8 is the distinguished output, where the Hamming numbers will appear in the correct order. M_1 and M_2 are *merge actors*, i.e., processes which take two ordered

[4]... or rather an earlier version from 2010: our 2013 paper could not have been influenced by a 2015 paper, nicely illustrating the very point of temporal constraints.

[5]also known as *regular numbers*

input sequences and merge them into an ordered output sequence. If presented with the same item in both streams, the output stream will only contain one copy of the element, so duplicates are removed. The actors X2, X3, and X5 multiply their inputs with 2, 3, and 5, respectively. The *sample-delay actors* S2, S3, S5 are used "to prime the pump": initially (i.e., before reading any input), they output the number 1 to get the loops started. Subsequently, they simply output whatever they receive as an input. By design, the Hamming workflows H_1 and H_3 define an infinite output stream, i.e., these processes can "run forever".

Figure 8 shows two provenance traces T_1 (*Fish*) and T_3 (*Sail*) for Hamming numbers $n \leq 1000$, corresponding to the workflow variants H_1 and H_3. To save space, the trace graphs show each invocation of a multiplication actor x2, x3, and x5 as a colored edge (green, blue, and red, respectively). By querying the trace graph, the answer relation can be obtained as a set of edges $d_1 \xrightarrow{p} d_2$, linking data items to each other, with the (implicit) label p denoting the actor invocations (multiplication factors) used. Note that while the workflow graphs in Figure 8 are cyclic, as expected, the trace graphs are acyclic. The trace-level retrospective provenance yields valuable information: In Figure 8(c) Hamming numbers n can be produced in many different ways (if n contains all three factors 2, 3, and 5, its in-degree is always three). As a result, the provenance graph T_1 is not a tree, but a DAG (directed acyclic graph). In contrast, in Figure 8(d) every Hamming number n is produced in one way only (there is a unique path from 1 to n), i.e., *without* creating unnecessary duplicates. Thus, unlike T_1, trace T_3 is a tree.

This example demonstrates another use of relating retrospective and prospective provenance, i.e., differences in the trace graphs T_1, T_3 can be used to explain the performance differences of the workflows H_1 and H_3 that generated them. Similarly, [KLS12] uses retrospective provenance to compare the efficiency of different variants of a transitive closure query. Other works making use of the relationships between prospective and retrospective provenance include [KSB+10, BML12, DBK+14, DBK+15, MBBL15].

3 Provenance in Databases

When comparing data provenance in workflows and in databases, the former is usually considered a form of *coarse-grained* provenance, while the latter is considered *fine-grained* provenance. Indeed, provenance from workflows often captures observables at the level of files read and written by workflows or scripts [MBBL15]. In contrast, provenance in databases aims to answer record-level questions, e.g., which tuples (rows) in the input tables contributed to a particular output tuple and how [CCT09]. Along another dimension, workflow provenance is sometimes called *black-box* provenance, whereas database provenance is considered *white-box* provenance [CCBD06, Tan07, Bow12]. This distinction is motivated by the fact that in workflows, the computational steps or actors are usually considered "black boxes" whose inner workings are not accessible or not relevant.[6] Conversely, as we shall see below, a database query can be considered a "white box", since its inner workings are readily available and analyzable [BT07, CCT09]. There are also approaches that combine workflow and database provenance, e.g., [ADD+11].

[6]However, workflow systems such as Kepler [LAB+06] support nested workflows, so it is possible to open these "grey boxes" [BL05, DBE+07, BCBDH08]. Similarly, fine-grained provenance from script-based workflows can be captured via profiling tools [MBC+14].

Database Provenance Questions. In the following, we consider the most widely-used and best studied database model, i.e., the relational model [AHV95]. But the basic principles usually also apply, *mutatis mutandis*, to other database models and queries, e.g., over semistructured (XML) data.[7] Consider a query answer $A = Q(D)$, i.e., an output table A resulting from the evaluation of a query Q on an input database D. Let $t \in A$ be a result tuple from the answer. In a database context, we would like to answer provenance questions such as:

What is the *lineage* of t, i.e., which specific subset(s) of tuples from the input D were used to produce t? Similarly, we might want to know *why* t is in the result and *how* exactly t was obtained from the tuples in its lineage. The notions of *lineage*, *why*-, and *how-provenance* (among others) have been formalized, studied in detail, and compared thoroughly [CCT09]. Before we illustrate these different notions with a running example, we first give a brief (and necessarily incomplete) overview of some key publications and milestones in database provenance.

3.1 A Brief History of Database Provenance

The idea of propagating annotations from sources through queries to results is at the core of many current database provenance approaches, but also had early precursors such as [WM+90], which proposed a model to carry along source attributions through queries. Another early approach which does not rely on annotations is described in [WS97]. Database research on provenance became mainstream through important, workflow-like applications in data warehouses [CWW00]. Data warehouses periodically retrieve and integrate information from multiple sources using extract-transform-load (ETL) scripts, and then make the integrated information readily available for online analytical processing (OLAP) [CD97]. In data warehousing and other information integration scenarios, it is often crucial to be able to trace the lineage of data from output tables back to the sources where the data originated. In this way, data quality problems can be detected, localized, and eventually resolved.

An influential paper by Buneman *et al.* [BKT01] developed the *why-provenance* model, refining another influential model by Cui *et al.* [CWW00] for tracing lineage in data warehousing applications. The *provenance semiring*[8] framework developed by Green *et al.* [GKT07] (and applied in a data sharing and information integration context [GKT10]) marks a milestone in provenance research, as it subsumes many earlier provenance models and embeds them in a single, unified framework.

All provenance models mentioned so far aim at explaining, at various levels of detail, *why* and *how* a query answer $t \in Q(D)$ came about. Thus, these database approaches aim to relate outputs back to the inputs on which they depend, i.e., at a high level, they resemble retrospective provenance models for workflows. The database community has also studied an intriguing new question, i.e., why is $t \notin Q(D)$? This *missing answer* problem is also known as *why-not provenance* [CJ09] and is an area of active research

[7] For example, [DT03, BGVK+06] show how XML queries can be reduced to relational queries.

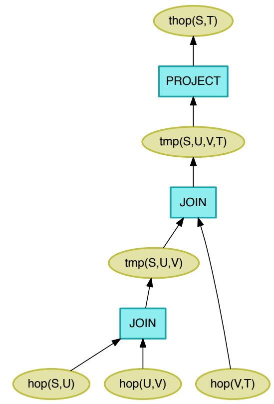
[8] In abstract algebra, a *semiring* is a structure $(K, +, \cdot, 0, 1)$ with binary operations "+" (addition) and "·" (multiplication) over an underlying set K satisfying, for all $x, y, z \in K$, these axioms: $x + y = y + x$; $x + 0 = 0 + x = x$; $x \cdot 1 = 1 \cdot x = x$; $x \cdot 0 = 0 \cdot x = 0$; $x \cdot (y + z) = x \cdot y + x \cdot z$; $(x + y) \cdot z = x \cdot z + y \cdot z$. If $x \cdot y = y \cdot x$, the semiring is *commutative*. Instead of $x \cdot y$ we can write xy.

$$\mathsf{thop}(S,T) \;:\!-\; \mathsf{hop}(S,U), \mathsf{hop}(U,V), \mathsf{hop}(V,T).$$

$$\begin{aligned}\mathsf{thop} := \pi_{S,T}(\quad & \rho_{T\to U}(\mathsf{hop}) \\ \bowtie \quad & \rho_{S\to U, T\to V}(\mathsf{hop}) \\ \bowtie \quad & \rho_{S\to V}(\mathsf{hop})\ )\end{aligned}$$

```
SELECT h1.S, h3.T
FROM   hop h1, hop h2, hop h3
WHERE  h1.T = h2.S
AND    h2.T = h3.S ;
```

(a)



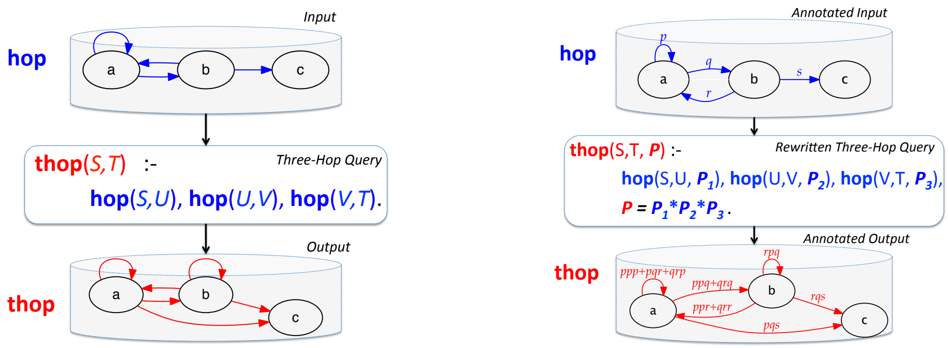
(b)

Figure 9: Three-Hop (thop) query [KG12], expressed in (a) Datalog (top), the Relational Algebra (middle), and SQL (bottom). (b) This query can also be considered a "mini-workflow" combining three copies of the hop relation via joins (denoted $\bowtie$ in the algebra), followed by a projection (denoted π) to yield the output relation thop.

[HH10, TC10, GP10, ADT11, KLZ13, BHT15, tCCST15]. We will return to this question briefly in Section 3.4.

The comprehensive survey by Cheney *et al.* [CCT09] classifies data provenance approaches into two broad categories called *lazy* and *eager*, respectively. In the lazy (or *non-annotation*) approaches, provenance is computed only on demand by examining and analyzing the input data D, the answers A, and the query Q. No changes are made to any of these. In contrast, the eager (or *annotation-based*) approaches use an annotated input database D' which is then evaluated using a rewritten "provenance-enabled" query Q' in order to obtain an answer table A' with provenance annotations. In the remainder of the paper, we focus on eager provenance approaches. In Appendix A we illustrate the exact nature of Q' and the provenance-annotated query answers A' via prototypical implementations of the running example discussed next.

Mixed forms combining aspects of eager and lazy approaches exist, e.g., [CWW00]. Several systems such as Perm [GMA13], GProM [AGR$^+$14], Ariadne [GEFT14], and PROV-Trace [SGB15] compute *provenance on demand* through provenance-enabled replay of operations. These systems therefore do not fit neatly into the two categories proposed in [CCT09]: On one hand, they appear lazy since provenance is not captured when evaluating a query but only later, if and when provenance is explicitly requested. On the other hand, the technique used for computing provenance is based on provenance-enabled queries that propagate annotations, i.e., the eager approach. The GProM system stands out since it is the first to support provenance tracking for *updates* (and transactions) based on MV-semirings, an extension of the semiring model with embedded multiversion history [AGK$^+$16].



(a) (b)

Figure 10: Three-Hop Example [KG12]: (a) The hop relation (blue) stores possible links in a network. The thop query (center) returns an output relation (red) consisting of all pairs (S, T) that can be connected by three hops $S \overset{\text{hop}}{\to} U \overset{\text{hop}}{\to} V \overset{\text{hop}}{\to} T$ (bottom). Typical provenance queries are: *Why* is thop(a, b) in the output, and *how* has thop(a, b) been derived from the input? (b) The provenance-annotated input is processed via a rewritten thop query, returning a provenance-annotated output that answers those questions: The provenance polynomial $p^2q + q^2r$ that annotates the thop(a, b) edge (bottom) means that there are two distinct ways from a to b using three hops: by using the p hop twice and the q hop once (p^2q), or alternatively, by q, r, and q again (q^2r).

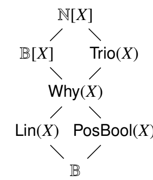
3.2 Running Example: The Three-Hop Query (thop)

Consider a database table hop that stores possible links between nodes in a network [KG12]. We might want to know which pairs of nodes are reachable with precisely three hops. Figure 9 shows this thop (Three-Hop) query in alternative but equivalent notations: as a Datalog query, a relational algebra query, and a SQL query. Finally, Figure 9(b) shows the same query in the form of a (relational algebra) operator tree. Using operator trees allows us to view a database query Q as a kind of workflow W_Q (or prospective provenance), and apply notions and techniques from Section 2. As mentioned before, the processing steps (actors) in workflows are usually considered black boxes. In contrast, in database queries, the semantics of query operators is completely known and available for analysis and query rewriting, making them white box actors that support fine-grained provenance capture. Now consider the thop query from Figure 9 applied to a concrete input database D as depicted in Figure 10(a). The input relation hop is shown as a directed graph (with blue edges). From this, the query computes a new graph (with red edges), shown at the bottom of Figure 10(a). Note that the hop input graph has no direct link from a to c, while the thop result graph has such as link.

Typical provenance queries are: *Why* is some tuple t in the output relation thop, and *how* has it been derived from the input relation hop? Consider the result tuple $t =$ (a, b) in thop. What is the *lineage* of t, i.e., what are the hop tuples that contributed to the derivation of the result thop(a, b)? Looking at the hop graph, we see that one can go from a to b using different edges from the input hop table, e.g., use the self-loop a $\to$ a twice, followed by the hop a $\to$ b, for a total of three hops. Another solution is to use a $\to$ b, then b $\to$ a, and

thop	S	T	Lin(X)	Why(X)	Trio(X)	$\mathbb{B}[X]$	$\mathbb{N}[X]$
	a	a	pqr	$p + pqr$	$p + 2pqr$	$p^3 + pqr$	$p^3 + 2pqr$
	a	b	pqr	$pq + qr$	$pq + qr$	$p^2q + q^2r$	$p^2q + q^2r$
	a	c	pqs	pqs	pqs	pqs	pqs
	b	a	pqr	$pr + qr$	$pr + qr$	$p^2r + qr^2$	$p^2r + qr^2$
	b	b	pqr	pqr	pqr	pqr	pqr
	b	c	qrs	qrs	qrs	qrs	qrs

(a)



(b)

Figure 11: Three-Hop Example (cont'd): (a) Provenance-annotated thop answer with five kinds of provenance. (b) The hierarchy among provenance models [KG12]: the finest-grain model $\mathbb{N}[X]$ subsumes other models such as Trio(X), Why(X), and Lin(X) below. For example, the Lin(X) model for thop(a, b) only states that the hop edges p, q, r are in the lineage, while the $\mathbb{N}[X]$ model states exactly *how* those edges need to be combined.

finally a $\to$ b one more time.

Figure 10(b) shows the same input database D_{hop} with a small but important modification: the edges in the hop relation are *annotated* with unique identifiers from an underlying set (or namespace) $X = \{p, q, r, s\}$. Thus, we can explain why thop(a, b) is in the answer simply by referring to the named edges: p, p, q is a three-hop from a to b, and q, r, q is another three-hop, i.e., a $\xrightarrow{p}$ a $\xrightarrow{p}$ a $\xrightarrow{q}$ b is the first solution, and a $\xrightarrow{q}$ b $\xrightarrow{r}$ a $\xrightarrow{q}$ b is the only other solution. A shorthand for the provenance-annotated result is thus "thop(a, b) : $p^2q + q^2r$". The *provenance polynomial* $p^2q + q^2r$ states why and how the query answer thop(a, b) was obtained from the hop input table. The addition "+" in the provenance polynomial corresponds to a logical disjunction ($\vee$) since there are two solutions to go from a to b using exactly three hop edges. Each solution consists of a product "·" of input tuples, corresponding to a logical conjunction ($\wedge$), i.e., $p \cdot p \cdot q$ and $q \cdot r \cdot q$. In the underlying provenance semiring [GKT07], the product and sum operations are commutative, hence the shorter polynomial representation $p^2q + q^2r$ can be used.

3.3 The Great Unification: Provenance Semirings

The representation of database provenance using abstract polynomials over annotated input databases was developed by Green *et al.* in [GKT07]; an introduction and overview is given in [KG12]. It is beyond the scope of this paper to elaborate on the details of that framework and its theoretical results (e.g., the "Fundamental Theorem"). However, using the running example, we can get a first idea of the elegance and power of the semiring approach. Figure 11(a) depicts the thop answer table with its six output tuples (corresponding to the six red thop edges in Figure 10). Each of the tuples in the provenance-annotated answer A' carries a provenance annotation which is obtained by executing a rewritten query Q' on an annotated input database D' (see also Appendix A). The most fine-grained provenance annotations are shown in the right-most column containing polynomials over the provenance semiring $\mathbb{N}[X]$. The other columns correspond to coarser provenance abstractions: e.g., $\mathbb{B}[X]$ is the semiring of Boolean provenance polynomials, Trio(X) is the provenance semiring used in the Trio system [BSHW06], while Why(X) and Lin(X) correspond to the *why-provenance* and *lineage* model in [BKT01] and [CWW00], respectively.

The lattice in Figure 11(b) shows the degree of "informedness" of the different provenance models (i.e., how "fine-grained" they are, relative to one another): as one moves down the lattice, provenance information becomes coarser. In our example, the $\mathbb{N}[X]$ provenance of $\mathsf{thop}(\mathsf{a},\mathsf{b})$ is p^2q+q^2r telling us (i) that there are exactly two ways to obtain the answer, and (ii) what those two ways are (one way uses the p edge twice and q once; the other uses q twice and r once). When looking instead at $\mathbb{B}[X]$, the coefficients are dropped from the polynomial, e.g., the provenance of $\mathsf{thop}(a,a)$ is p^3+2pqr in $\mathbb{N}[X]$, but becomes p^3+pqr in $\mathbb{B}[X]$. Similarly, in $\mathsf{Trio}(X)$, exponents are dropped, in $\mathsf{Why}(X)$ coefficients *and* exponents are dropped, and in $\mathsf{Lin}(X)$ only the (flat) union of tuples pqr remains to describe the lineage of $\mathsf{thop}(a,a)$, i.e., these three edges were used in the derivation, but it is not stated *how* they need to be put together to derive a three-hop from a to a.

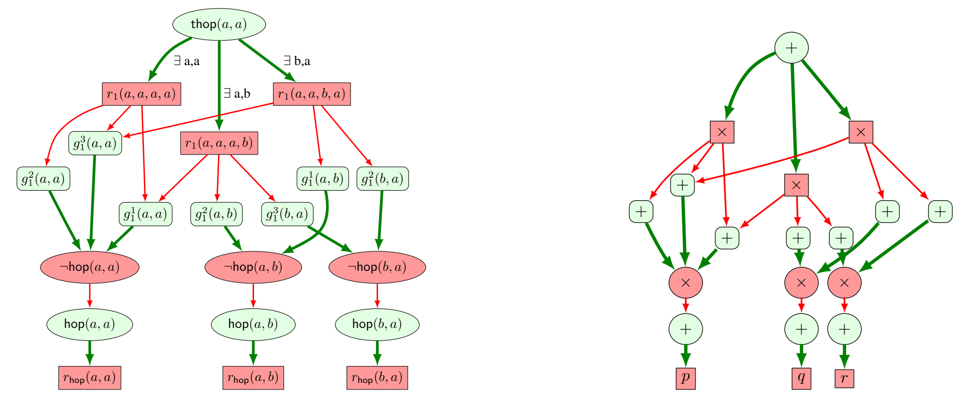
The "Fundamental Theorem" [KG12] intuitively states that for positive relational algebra queries one can swap the order of query evaluation and application of a semiring homorphism. For example, consider an input database with annotations $p,q,r,\ldots$ that represent Boolean variables that can be either *true* or *false*, indicating whether the so-annoted tuple is or isn't true in the modeled world. In order to explore the answers to a query Q in different possible worlds (i.e., under different truth assignments to the Booleans), we could run the query Q once for each such possible world. Alternatively, we can execute the provenance-enabled query Q' once (and for all) to obtain provenance polynomials in $\mathbb{N}[X]$ as depicted in Figure 11(a). To obtain the different possible worlds, we then just reinterpret the provenance-polynomials as Boolean expressions ("$\cdot$" as "$\wedge$" and "+" as "$\vee$") and simplify those Boolean expressions. Both routes (Boolean assignment followed by query evaluation or vice versa) will yield the same result.

Appendix A contains another example, where the input annotations represent tuple cardinalities in the relational model with multiset (bag) semantics. We can evaluate the query under the bag semantics to obtain the result cardinalities (Fig. 15(c) and 15(d)). Alternatively, we can "plug in" the input cardinalities into the abstract provenance polynomials in Fig. 15(b) and then evaluate those polynomials to arrive at the same numbers as in Fig. 15(d).

3.4 Unifying Why and Why-Not Provenance through Games

The elegant and powerful provenance semiring approach by Green *et al.* [GKT07, KG12] subsumes and situates many earlier database provenance models. However, one shortcoming of those approaches is that they are limited to *positive* queries only, i.e., they cannot handle queries with negation. On the other hand, if a provenance approach can be devised that can answer queries with negation, then such an approach would also solve the missing answers or why-not provenance problem: Asking why is $\mathsf{thop}(\mathsf{c},\mathsf{a})$ *not* in the answer is then equivalent to asking: why is $\neg\mathsf{thop}(\mathsf{c},\mathsf{a})$ true over the given database. Figure 12(a) depicts a solved *provenance game* for $\mathsf{thop}(\mathsf{a},\mathsf{a})$. This approach was developed by Köhler *et al.* [KLZ13] and contains the provenance semiring approach as a special case, see Figure 12(b). The key idea is to view query evaluation $A = Q(D)$ as a game between two players who argue whether or not tuple $t \in A$.[9] The game can be defined in such a way

[9]Query evaluation games [Hod13] have been considered before, e.g., by Hintikka [Hin96]. However, the idea of using games for provenance was inspired more recently by revisiting the game normal form [FKL97] for well-founded Datalog.



(a) (b)

Figure 12: Why (and how) is $\mathsf{thop}(\mathsf{a},\mathsf{a})$ in the query answer? (a) The solved provenance game [KLZ13] shows that one can find three different instances of the thop Datalog rule that are satisfied. (b) The solved game DAG can be abstracted and expanded into a tree to yield the provenance polynomial for $\mathsf{thop}(\mathsf{a},\mathsf{a})$: $p^3 + 2pqr$. (Product "$\cdot$" shown as "$\times$" here.)

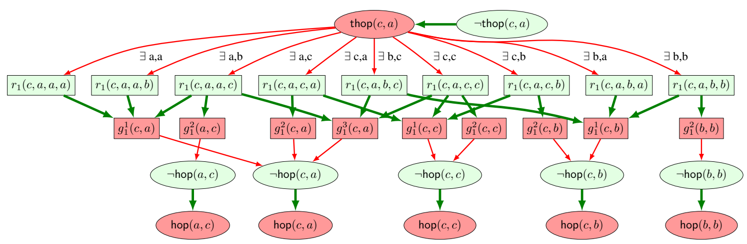


Figure 13: Why-not provenance for $\mathsf{thop}(\mathsf{c},\mathsf{a})$ using provenance games [KLZ13]. The graph enumerates all (failed) attempts to prove $\mathsf{thop}(\mathsf{c},\mathsf{a})$ using the thop query over the given hop database. This structure can also be used to propose changes to the database such that $\mathsf{thopc}(\mathsf{c},\mathsf{a})$ will be in the answer.

that whoever is right about the claim can force a win [KLZ13]. Then the provenance (or *justifications*) for a claim about $t \in A$ can be obtained from a solved game graph such as the one in Figure 12(a).

A key advantage of this approach is that it treats why and why-not provenance uniformly: Figure 13 depicts a solved query evaluation game establishing why $\mathsf{thop}(\mathsf{c},\mathsf{a})$ is *not* in the answer. The solved game graph contains the equivalent of all failed proof attempts using the rules of the query (corresponding to failed SLD(NF) trees [AD94]) and can be used, e.g., to determine how the given database can be "fixed" so that $\mathsf{thop}(\mathsf{c},\mathsf{a})$ becomes true after all.

4 Conclusions

Provenance is a flourishing research topic in many subdisciplines of computer science. The scientific workflow community has contributed to the development of the Open Provenance Model (OPM) and its W3C successor PROV [Mor10]. As described in this chapter, two main forms of provenance can be distinguished in workflows, i.e., prospective and retrospective provenance. When combined in a single model of provenance (possibly enriched with temporal information), powerful provenance queries can be answered. The database community has developed another set of provenance models which abstract tuple derivations through relational queries (or Datalog rules). The provenance semiring model introduced by Green *et al.* [GKT07] elegantly subsumes many earlier provenance models for positive queries. Why-not (or missing answer) provenance is an active area of research.

In this brief tour, many interesting topics in workflow provenance and database provenance could not be covered; e.g., see [MPB10, MLB+10, ZL10, KRZ+11] and [MGMS10, SB15], respectively. For overviews and surveys on provenance and workflow see, e.g., [DBE+07, Mor10, Bow12, CVDK+12]. For further reading on provenance in databases, [CCT09] provides an excellent starting point.

Acknowledgments. This work was supported in part by NSF grants ACI-1430508, DBI-{1147273, 1356751}, IIS-1118088, and SMA-1439603. With special thanks to Shawn Bowers, Timothy McPhillips, Manish K. Anand, Víctor Cuevas-Vicenttín, Saumen Dey, Lei Dou, Sven Köhler, Sean Riddle, and Daniel Zinn for fruitful years of collaboration on scientific workflows and database provenance. Also special thanks to Boris Glavic for comments on an earlier draft of this paper and for his collaboration on and implementation of games for why-not provenance.

References

[ABL09] M. K. Anand, S. Bowers, and B. Ludäscher. A navigation model for exploring scientific workflow provenance graphs. In *4th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2009.

[ABL10a] M. K. Anand, S. Bowers, and B. Ludäscher. Provenance Browser: Displaying and Querying Scientific Workflow Provenance Graphs. In *International Conference on Data Engineering (ICDE)*, pp. 1201–1204. IEEE, 2010.

[ABL10b] M. K. Anand, S. Bowers, and B. Ludäscher. Techniques for efficiently querying scientific workflow provenance graphs. *EDBT*, 10:287–298, 2010.

[ABL12] M. K. Anand, S. Bowers, and B. Ludäscher. Database support for exploring scientific workflow provenance graphs. In *Scientific and Statistical Database Management*, pp. 343–360. Springer Berlin Heidelberg, 2012.

[ABML09] M. K. Anand, S. Bowers, T. McPhillips, and B. Ludäscher. Efficient provenance storage over nested data collections. In *International Conference on Extending Database Technology (EDBT)*, pp. 958–969. ACM, 2009.

[AD94] K. R. Apt and K. Doets. A new definition of SLDNF-resolution. *The Journal of Logic Programming*, 18(2):177–190, 1994.

[ADD+11] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. Putting lipstick on pig: Enabling database-style workflow provenance. *Proc. of the VLDB Endowment*, 5(4):346–357, 2011.

[ADT11] Y. Amsterdamer, D. Deutch, and V. Tannen. On the Limitations of Provenance for Queries with Difference. In *TaPP*, 2011.

[AGK+16] B. Arab, D. Gawlick, V. Krishnaswamy, V. Radhakrishnan, and B. Glavic. Formal Foundations of Reenactment and Transaction Provenance. Technical Report IIT/CS-DB-2016-01, Illinois Institute of Technology, 2016.

[AGR+14] B. Arab, D. Gawlick, V. Radhakrishnan, H. Guo, and B. Glavic. A Generic Provenance Middleware for Queries, Updates, and Transactions. In *6th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2014.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[BCBDH08] O. Biton, S. Cohen-Boulakia, S. B. Davidson, and C. S. Hara. Querying and managing provenance through user views in scientific workflows. In *Intl. Conf. on Data Engineering (ICDE)*, pp. 1072–1081. IEEE, 2008.

[BGVK+06] P. Boncz, T. Grust, M. Van Keulen, S. Manegold, J. Rittinger, and J. Teubner. MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In *SIGMOD*, pp. 479–490. ACM, 2006.

[BHT15] N. Bidoit, M. Herschel, and K. Tzompanaki. EFQ: Why-not answer polynomials in action. *Proceedings of the VLDB Endowment*, 8(12):1980–1983, 2015.

[BKT01] P. Buneman, S. Khanna, and W.-C. Tan. Why and Where: A Characterization of Data Provenance. In *ICDT*, pp. 316–330. Springer, 2001.

[BL05] S. Bowers and B. Ludäscher. Actor-oriented design of scientific workflows. In *Conceptual Modeling (ER)*, LNCS 3716, pp. 369–384. Springer, 2005.

[BML12] S. Bowers, T. McPhillips, and B. Ludäscher. Declarative rules for inferring fine-grained data provenance from scientific workflow execution traces. In *Intl. Provenance and Annotation Workshop (IPAW)*, pp. 82–96. Springer, 2012.

[BMR+08] S. Bowers, T. McPhillips, S. Riddle, M. K. Anand, and B. Ludäscher. Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life. In *Provenance and Annotation of Data and Processes (IPAW)*, pp. 70–77. Springer Berlin Heidelberg, 2008.

[Bow12] S. Bowers. Scientific workflow, provenance, and data modeling challenges and approaches. *Journal on Data Semantics*, 1(1):19–30, 2012.

[BSHW06] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proceedings of the 32nd international conference on Very large data bases*, pp. 953–964. VLDB Endowment, 2006.

[BT07] P. Buneman and W.-C. Tan. Provenance in Databases (Tutorial Outline). In *SIGMOD*, pp. 1171–1173. ACM, 2007.

[CCBD06] S. Cohen, S. Cohen-Boulakia, and S. Davidson. Towards a model of provenance and user views in scientific workflows. In *Data Integration in the Life Sciences (DILS)*, pp. 264–279. Springer, 2006.

[CCT09] J. Cheney, L. Chiticariu, and W. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[CD97] S. Chaudhuri and U. Dayal. Data warehousing and OLAP for decision support. *ACM Sigmod Record*, 26(2):507–508, 1997.

[CJ09] A. Chapman and H. Jagadish. Why not? In *SIGMOD*, pp. 523–534. ACM, 2009.

[CJR08] A. P. Chapman, H. V. Jagadish, and P. Ramanan. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 993–1006. ACM, 2008.

[CVDK+12] V. Cuevas-Vicenttín, S. Dey, S. Köhler, S. Riddle, and B. Ludäscher. Scientific Workflows and Provenance: Introduction and Research Opportunities. *Datenbank-Spektrum*, 12(3):193–203, 2012.

[CWW00] Y. Cui, J. Widom, and J. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems (TODS)*, 25(2):179–227, 2000.

[DBE+07] 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.

[DBK+14] S. Dey, K. Belhajjame, D. Koop, T. Song, P. Missier, and B. Ludäscher. UP & DOWN: Improving Provenance Precision by Combining Workflow-and Trace-Level Information. In *6th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, Cologne, Germany, 2014.

[DBK+15] S. Dey, K. Belhajjame, D. Koop, M. Raul, and B. Ludäscher. Linking Prospective and Retrospective Provenance for Scripts. In *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, Edinburgh, Scotland, 2015.

[DCM+12] L. Dou, G. Cao, P. J. Morris, R. A. Morris, B. Ludäscher, J. A. Macklin, and J. Hanken. Kurator: A Kepler Package for Data Curation Workflows. *Procedia CS: Proceedings Intl. Conf. on Computational Science (ICSS)*, 9:1614–1619, 2012. demo video at http://youtu.be/DEkPbvLsud0.

[Dij81] E. W. Dijkstra. Hamming's exercise in SASL, 1981. EWD-792.

[DKBL12] S. Dey, S. Köhler, S. Bowers, and B. Ludäscher. Datalog as a Lingua Franca for Provenance Querying and Reasoning. In *Workshop on the Theory and Practice of Provenance (TaPP)*, Boston, MA, 2012.

[Dob73] T. Dobzhansky. Nothing in Biology Makes Sense except in the Light of Evolution. *The American Biology Teacher*, 35(3):125–129, 1973.

[DRL13] S. Dey, S. Riddle, and B. Ludäscher. Provenance Analyzer: Exploring Provenance Semantics with Logic Rules. In *5th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2013.

[DT03] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *Intl. Conference on Database Theory (ICDT)*, pp. 225–241. Springer, 2003.

[EMN14] I. Eisenman, W. N. Meier, and J. R. Norris. A spurious jump in the satellite record: has Antarctic sea ice expansion been overestimated? *The Cryosphere*, 8(4):1289–1296, 2014.

[FKL97] J. Flum, M. Kubierschky, and B. Ludäscher. Total and Partial Well-Founded Datalog Coincide. In *ICDT*, pp. 113–124, 1997.

[GCI15] GCIS. Global Change Information System. http://data.globalchange.gov/, 2015.

[GEFT14] B. Glavic, K. S. Esmaili, P. M. Fischer, and N. Tatbul. Efficient stream provenance via operator instrumentation. *ACM Transactions on Internet Technology (TOIT)*, 14(1):7, 2014.

[GG11] D. Garijo and Y. Gil. A new approach for publishing workflows: abstractions, standards, and linked data. In *6th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2011.

[GKT07] T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pp. 31–40, 2007.

[GKT10] T. J. Green, G. Karvounarakis, and Z. G. I. V. Tannen. Provenance in ORCHESTRA. *Bulletin of the Technical Committee on Data Engineering*, 33(3):9–16, September 2010. IEEE Computer Society.

[GMA13] B. Glavic, R. J. Miller, and G. Alonso. Using SQL for Efficient Generation and Querying of Provenance Information. In *In Search of Elegance in the Theory and Practice of Computation. Essays Dedicated to Peter Buneman*, volume 8000 of *LNCS*, pp. 291–320. Springer, 2013.

[GP10] F. Geerts and A. Poggi. On database query languages for k-relations. *Journal of Applied Logic*, 8(2):173–185, 2010.

[HA08] T. Heinis and G. Alonso. Efficient lineage tracking for scientific workflows. In *SIGMOD*, pp. 1007–1018. ACM, 2008.

[HDD+15] D. J. Hills, R. R. Downs, R. Duerr, J. C. Goldstein, M. A. Parsons, and H. K. Ramapriyan. The importance of data set provenance for science. *EOS*, 96, 2015. doi:10.1029/2015EO040557.

[Hem88] D. Hemmendinger. The "Hamming problem" in Prolog. *ACM SIGPLAN Notices*, 23(4):81–86, 1988.

[HH10] M. Herschel and M. A. Hernández. Explaining missing answers to SPJUA queries. *Proceedings of the VLDB Endowment*, 3(1-2):185–196, 2010.

[Hin96] J. Hintikka. *The Principles of Mathematics Revisited*. Cambridge University Press, 1996.

[Hod13] W. Hodges. Logic and Games. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. http://plato.stanford.edu/entries/logic-games/, March 2013.

[HTT09] T. Hey, S. Tansley, and K. Tolle, editors. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009.

[KG12] G. Karvounarakis and T. J. Green. Semiring-annotated data: queries and provenance. *ACM SIGMOD Record*, 41(3):5–14, 2012.

[KLS12] S. Köhler, B. Ludäscher, and Y. Smaragdakis. Declarative datalog debugging for mere mortals. In *Datalog in Academia and Industry*, pp. 111–122. Springer Berlin Heidelberg, 2012.

[KLZ13] S. Köhler, B. Ludäscher, and D. Zinn. First-Order Provenance Games. In *In Search of Elegance in the Theory and Practice of Computation. Essays Dedicated to Peter Buneman*, volume 8000 of *LNCS*, pp. 382–399. Springer, 2013.

[KMB15] N. Kwasnikowska, L. Moreau, and J. V. D. Bussche. A Formal Account of the Open Provenance Model. *ACM Transactions on the Web (TWEB)*, 9(2):10:1–10:44, 2015.

[KRZ+11] S. Köhler, S. Riddle, D. Zinn, T. McPhillips, and B. Ludäscher. Improving workflow fault tolerance through provenance-based recovery. In *Scientific and Statistical Database Management*, pp. 207–224. Springer Berlin Heidelberg, 2011.

[KSB+10] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva. Bridging workflow and data provenance using strong links. In M. Gertz and B. Ludäscher, editors, *Scientific and statistical database management (SSDBM)*, LNCS 6187, pp. 397–415. Springer, 2010.

[LAB+06] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[LAB+09] B. Ludäscher, I. Altintas, S. Bowers, J. Cummings, T. Critchlow, E. Deelman, D. D. Roure, J. Freire, C. Goble, M. Jones, S. Klasky, T. McPhillips, N. Podhorszki, C. Silva, I. Taylor, and M. Vouk. Scientific Process Automation and Workflow Management. In A. Shoshani and D. Rotem, editors, *Scientific Data Management*. Chapman & Hall/CRC, 2009.

[LBM09] B. Ludäscher, S. Bowers, and T. McPhillips. Scientific Workflows. In T. Özsu and L. Liu, editors, *Encyclopedia of Database Systems*. Springer, 2009.

[MBBL15] T. McPhillips, S. Bowers, K. Belhajjame, and B. Ludäscher. Retrospective provenance without a runtime provenance recorder. In *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, Edinburgh, Scotland, 2015.

[MBC+14] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noWorkflow: Capturing and analyzing provenance of scripts. In *Provenance and Annotation of Data and Processes (IPAW)*, pp. 71–83. Springer, 2014.

[MBZL09] T. McPhillips, S. Bowers, D. Zinn, and B. Ludäscher. Scientific workflow design for mere mortals. *Future Generation Computer Systems*, 25(5):541–551, 2009.

[MCF+11] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche. The open provenance model core specification (v1. 1). *Future Generation Computer Systems*, 27(6):743–756, 2011.

[MDB+13] P. Missier, S. Dey, K. Belhajjame, V. Cuevas-Vicenttín, and B. Ludäscher. D-PROV: Extending the PROV Provenance Model with Workflow Structure. In *5th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2013.

[MFF+08] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The Open Provenance Model: An Overview. In *Provenance and Annotation of Data and Processes*, pp. 323–326. Springer, 2008.

[MGMS10] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proceedings of the VLDB Endowment*, 4(1):34–45, 2010.

[MLA+08] L. Moreau, B. Ludäscher, I. Altintas, R. S. Barga, S. Bowers, S. Callahan, G. Chin, B. Clifford, S. Cohen, S. Cohen-Boulakia, S. Davidson, E. Deelman, L. Digiampietri, I. Foster, J. Freire, J. Frew, J. Futrelle, T. Gibson, Y. Gil, C. Goble, J. Golbeck, P. Groth, D. A. Holland, S. Jiang, J. Kim, D. Koop, A. Krenek, T. McPhillips, G. Mehta, S. Miles, D. Metzger, S. Munroe, J. Myers, B. Plale, N. Podhorszki, V. Ratnakar, E. Santos, C. Scheidegger, K. Schuchardt, M. Seltzer, Y. L. Simmhan, C. Silva, P. Slaughter, E. Stephan, R. Stevens, D. Turi, H. Vo, M. Wilde, J. Zhao, and Y. Zhao. Special Issue: The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*, 20(5):409–418, 2008.

[MLB+10] P. Missier, B. Ludäscher, S. Bowers, S. Dey, A. Sarkar, B. Shrestha, I. Altintas, M. K. Anand, and C. Goble. Linking multiple workflow provenance traces for interoperable collaborative science. In *5th Workshop on Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 2010.

[MMB+12] L. Moreau, P. Missier, K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, S. Miles, J. Myers, S. Sahoo, and C. Tilmes. The PROV Data Model. W3C Technical Report, https://www.w3.org/TR/prov-dm/, 2012.

[Mor10] L. Moreau. The foundations for provenance on the web. *Foundations and Trends in Web Science*, 2(2–3):99–241, 2010.

[MPB10] P. Missier, N. W. Paton, and K. Belhajjame. Fine-grained and efficient lineage querying of collection-based workflow provenance. In *EDBT*, pp. 299–310, 2010.

[MRY14] J. M. Melillo, T. T. Richmond, and G. W. Yohe, editors. *Climate Change Impacts in the United States: The Third National Climate Assessment*. U.S. Global Change Research Program, 2014. doi:10.7930/J0Z31WJ2.

[MZH+08] M. E. Mann, Z. Zhang, M. K. Hughes, R. S. Bradley, S. K. Miller, S. Rutherford, and F. Ni. Proxy-based reconstructions of hemispheric and global surface temperature variations over the past two millennia. *Proceedings of the National Academy of Sciences*, 105(36):13252–13257, 2008.

[PMGF15] Q. Pham, T. Malik, B. Glavic, and I. Foster. LDV: Light-weight Database Virtualization. In *International Conference on Data Engineering (ICDE)*, pp. 1179–1190, 2015.

[Sad13] S. Sadiq. *Handbook of data quality*. Springer, 2013.

[SB15] B. Salimi and L. Bertossi. From Causes for Database Queries to Repairs and Model-Based Diagnosis and Back. In *18th International Conference on Database Theory (ICDT)*, volume 31, pp. 342–362. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

[SGB15] M. Stamatogiannakis, P. Groth, and H. Bos. Decoupling Provenance Capture and Analysis from Execution. In *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2015.

[Ste13] L. Stevens. Texas Summer 2011: Record Heat and Drought. GCIS metadata record with provenance, 2013. accessed 2015-12-12.

[Tan07] W.-C. Tan. Provenance in Databases: Past, Current, and Future. *IEEE Data Engineering Bulletin*, 30(4):3–12, 2007.

[TC10] Q. T. Tran and C.-Y. Chan. How to ConQueR Why-Not Questions. In *SIGMOD*, pp. 15–26. ACM, 2010.

[tCCST15] B. ten Cate, C. Civili, E. Sherkhonov, and W.-C. Tan. High-Level Why-Not Explanations using Ontologies. In *ACM Symposium on Principles of Database Systems (PODS)*, pp. 31–43. ACM, 2015.

[TFM+13] C. Tilmes, P. Fox, X.-L. Ma, D. L. McGuinness, A. P. Privette, A. Smith, A. Waple, S. Zednik, and J. G. Zheng. Provenance representation for the national climate assessment in the global change information system. *Geoscience and Remote Sensing, IEEE Transactions on*, 51(11):5160–5168, 2013.

[Wed11] M. J. Wedel. A monument of inefficiency: The presumed course of the recurrent laryngeal nerve in sauropod dinosaurs. *Acta Palaeontologica Polonica*, 57(2):251–256, 2011.

[WM+90] Y. R. Wang, S. E. Madnick, et al. A polygen model for heterogeneous database systems: The source tagging perspective. In *VLDB*, volume 90, pp. 519–538, 1990.

[WS97] A. Woodruff and M. Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Intl. Conference on Data Engineering (ICDE)*, pp. 91–102. IEEE, 1997.

[ZL10] D. Zinn and B. Ludäscher. Abstract provenance graphs: anticipating and exploiting schema-level data provenance. In *Provenance and Annotation of Data and Processes*, pp. 206–215. Springer Berlin Heidelberg, 2010.

A Query Rewriting for Provenance Annotations

```
% hop(X,Y) relation with unique tuple-ids p,q,r,s:
hop(a,a, p).
hop(a,b, q).
hop(b,a, r).
hop(b,c, s).

% Rewritten Three-Hop Query:
thop_aux(X,Y, P) :-
        hop(X,U, P1), hop(U,V, P2), hop(V,Y, P3),
        P = P1*P2*P3.

% For each X,Y pair, aggregate all provenance annotations.
thop(X,Y, Ps) :-
        bagof( P, thop_aux(X,Y, P), Ps ).

% Backtracking loop to generate all thop(X,Y) with provenance:
:-      thop(X,Y, Ps), format("thop(~w,~w) : ~w~n", [X,Y,Ps]), fail
    ;   true.
```

(a)

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.0.2)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [thop].
thop(a,a) : [p*p*p,p*q*r,q*r*p]
thop(a,b) : [p*p*q,q*r*q]
thop(a,c) : [p*q*s]
thop(b,a) : [r*p*p,r*q*r]
thop(b,b) : [r*p*q]
thop(b,c) : [r*q*s]
% thop compiled 0.00 sec, 8 clauses
true.

?-
```

(b)

```
:- use_module(library(lists)). % for arithmetic list sum

% hop(X,Y) multiset with cardinalities:
hop(a,a, 1).
hop(a,b, 4).
hop(b,a, 2).
hop(b,c, 3).

% Rewritten Three-Hop Query:
thop_aux(X,Y,  N) :-
        hop(X,U, N1), hop(U,V, N2), hop(V,Y, N3),
        N is N1*N2*N3.

% For each X,Y pair, aggregate all provenance annotations:
thop(X,Y, Ns) :-
        bagof( N , thop_aux(X,Y,N), Ns ).

% Backtracking loop to generate all thop(X,Y) with provenance:
:-      thop(X,Y, Ns), sumlist(Ns,S), % arithmetic list sum
        format('thop(~w,~w) : ~w~n', [X,Y,S]), fail
    ;   true.
```

(c)

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.0.2)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [thop2].
%   library(error) compiled into error 0.00 sec, 79 clauses
%  library(lists) compiled into lists 0.00 sec, 179 clauses
thop(a,a) : 17
thop(a,b) : 36
thop(a,c) : 12
thop(b,a) : 18
thop(b,b) : 8
thop(b,c) : 24
% thop2 compiled 0.01 sec, 190 clauses
true.

?-
```

(d)

Figure 14: Three-Hop Example [KG12] prototypically implemented in SWI-Prolog: (a) The input relation hop is annotated with unique tuple-ids. The rewritten view thop_aux adds the symbolic product $\mathsf{P} = \mathsf{P}_1 \cdot \mathsf{P}_2 \cdot \mathsf{P}_3$, combining the provenance annotations P_i of all hop tuples being joined. Aggregation with bagof/3 (instead of setof/3) is used to collect all provenance. (b) Running the code from (a) generates the provenance polynomials. (c) Similar to (a) but now hop is a multiset with cardinality annotations. The provenance of the thop result is calculated as the sum of the *arithmetic* product of the input cardinalities. (d) Running the code from (c) generates the result cardinalities.

The key ideas behind the query rewriting in the semiring annotation approach [GKT07] can be nicely illustrated using some simple prototypical implementations.[10] Figure 14 depicts two variants of the three-hop query [KG12] used earlier in the paper. The first variant (Figure 14(a)) uses unique tuple-ids and a symbolic representation of the product operation in the $\mathbb{N}[X]$ semiring. Lists of such products are used to represent the sum of products form in Figure 14(b). In Figure 14(c) the same query is used, but now hop represents a multiset (bag semantics), so tuples are annotated with cardinalities (how many times a tuple is in the multiset). The resulting cardinalities in the thop result relation are obtained by computing the sum of the arithmetic products of the cardinalities of hop tuples being joined to obtain the annotated thop tuples. The use of bag semantics (via the built-in aggregation predicate bagof/3, rather than setof/3) is essential to obtain the correct cardinalities.

[10]The example code is available from github.com/idaks/tour-de-provenance

```
CREATE TABLE hop (S text, T text, P text);

INSERT INTO hop VALUES ("a","a", "p");
INSERT INTO hop VALUES ("a","b", "q");
INSERT INTO hop VALUES ("b","a", "r");
INSERT INTO hop VALUES ("b","c", "s");

CREATE VIEW thop AS
    SELECT h1.S, h3.T, h1.P||'*'||h2.P||'*'||h3.P AS P
    FROM   hop h1, hop h2, hop h3
    WHERE  h1.T = h2.S AND h2.T = h3.S ;

SELECT   S, T, group_concat(P, ' + ')
FROM     thop
GROUP BY S, T;
```

(a)

```
$ sqlite3 -init thop.sql
-- Loading resources from thop.sql
S          T          group_concat(P, ' + ')
---------- ---------- ----------------------
a          a          p*p*p + p*q*r + q*r*p
a          b          p*p*q + q*r*q
a          c          p*q*s
b          a          r*p*p + r*q*r
b          b          r*p*q
b          c          r*q*s

SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
sqlite>
```

(b)

```
CREATE TABLE hop (S text, T text, P number);

INSERT INTO hop VALUES ("a","a", 1);
INSERT INTO hop VALUES ("a","b", 4);
INSERT INTO hop VALUES ("b","a", 2);
INSERT INTO hop VALUES ("b","c", 3);

CREATE VIEW thop AS
    SELECT h1.S, h3.T, h1.P * h2.P * h3.P AS P
    FROM   hop h1, hop h2, hop h3
    WHERE  h1.T = h2.S AND h2.T = h3.S ;

SELECT   S, T, sum(P)
FROM     thop
GROUP BY S, T;
```

(c)

```
$ sqlite3 -init thop2.sql
-- Loading resources from thop2.sql
S          T          sum(P)
---------- ---------- ----------
a          a          17
a          b          36
a          c          12
b          a          18
b          b          8
b          c          24

SQLite version 3.8.11.1 2015-07-29 20:00:57
Enter ".help" for usage hints.
sqlite>
```

(d)

Figure 15: Three-Hop Example [KG12] prototypically implemented in SQLite via query rewritings: (a) the rewritten view `thop` adds a column that symbolically"multiplies" the provenance of the `hop` tuples being joined; (b) running the aggregation query from (a) yields the provenance polynomials from $\mathbb{N}[X]$; (c) variant similar to (a) but for bag semantics; (d) running the aggregation from (c) yields the expected multiplicities.

In Figure 15 the same thop query with provenance is implemented in SQLite, again first using provenance polynomials over the $\mathbb{N}[X]$ semiring (using symbolic tuple-ids, represented as strings). The second variant in Figure 15(c) and 15(d) uses multiset semantics where tuple cardinalities are represented numerically in an additional column. The result cardinalities are then obtained via a summation over the (arithmetic) products of thop annotations.