Funded by the
European Union

Data Provenance

Silviu Maniu

. . . with help from Pierre Senellart, Yann Ramusat, Aryak Sen

April 28th, 2026

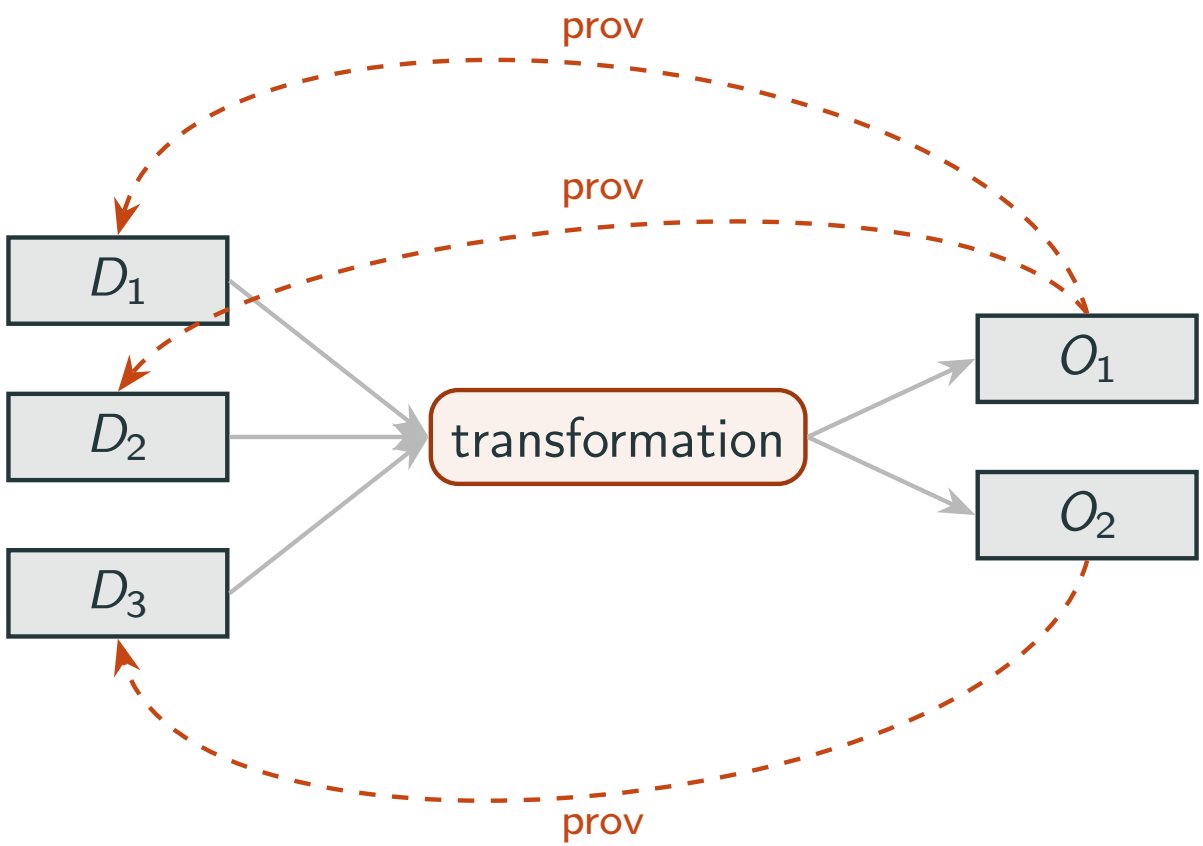
MDD & ARMADA Summer School 2026 **Cargèse, Corsica, France**

Introduction

Provenance

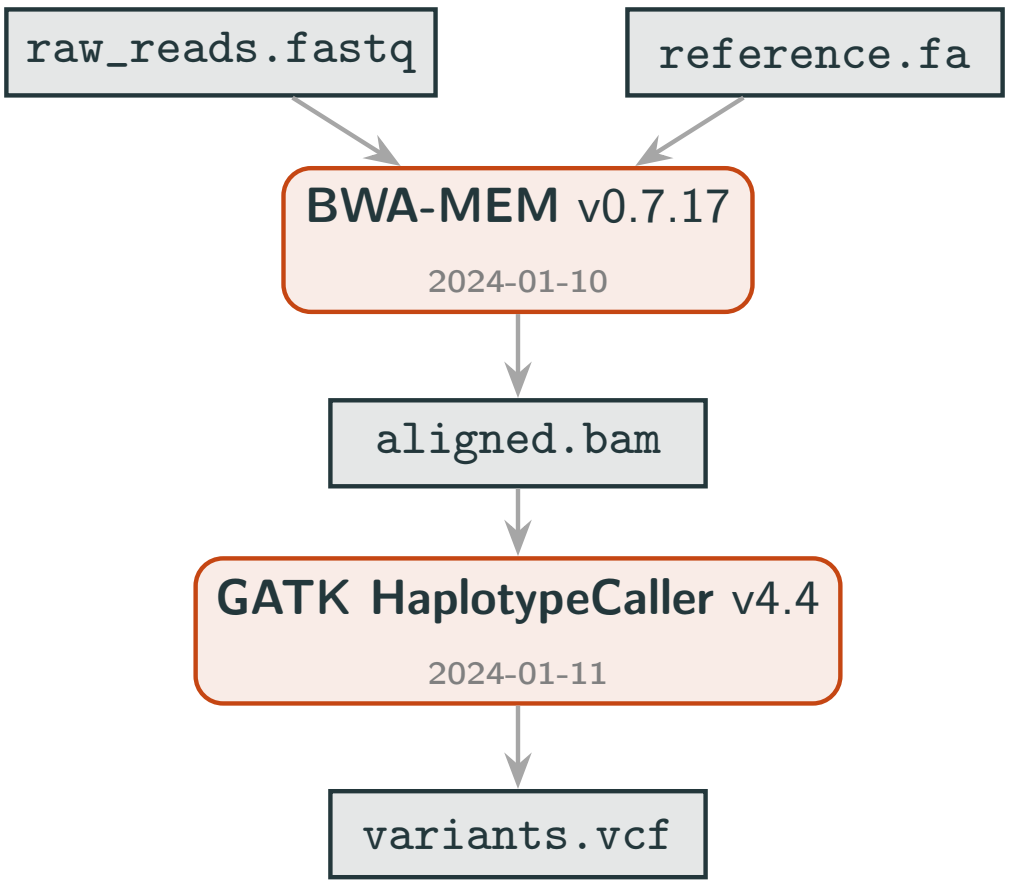
Provenance: information about the origin of data and the process by which it was produced

In data management, we care about data transformations.



Types of provenance: Workflow provenance

Workflow provenance: history and lineage of data in a *computational workflow* [Davidson et al., 2007]



Types of provenance: Workflow provenance

Workflow provenance: history and lineage of data in a *computational workflow* [Davidson et al., 2007]

- uniquely identifies the data (dataset granularity) used and produced
- documents actions used (tool, date, version, etc.)
- results in a directed acyclic graph structure (dataflow)
- formalized in the W3C PROV standard [W3C, 2024]

Types of provenance: Fine-grained provenance

Fine-grained provenance: information about data *fragments* that contributed to the production of a given result [Green et al., 2007]

name	**city**	
John	N. York	t_1
Paul	N. York	t_2
Dave	Paris	t_3
Ellen	Berlin	t_4
Magdalen	Paris	t_5
Nancy	Paris	t_6
Susan	Berlin	t_7

$\xrightarrow{\pi_{\text{city}}}$

city	
N. York	$\{t_1\}, \{t_2\}$
Paris	$\{t_3\}, \{t_5\}, \{t_6\}$
Berlin	$\{t_4\}, \{t_7\}$

Types of provenance: Fine-grained provenance

In databases, fine-grained provenance can be used to:

- understand the results of a query by tracing back to the input data
- understand how the query results react to changes in the data
- debug queries by identifying the source of errors in the results

Why-provenance

Why provenance – which input data contributed to the production of a given result:

- linked to the concept of data lineage in databases
- one way to represent is via **witness sets** (of input tuples)

name	**city**	
John	N. York	t_1
Paul	N. York	t_2
Dave	Paris	t_3
Ellen	Berlin	t_4
Magdalen	Paris	t_5
Nancy	Paris	t_6
Susan	Berlin	t_7

$\xrightarrow{\pi_{\text{city}}}$

city	
N. York	$\{t_1\}, \{t_2\}$
Paris	$\{t_3\}, \{t_5\}, \{t_6\}$
Berlin	$\{t_4\}, \{t_7\}$

How-provenance

How provenance – how the result has been computed, e.g., which operations were applied to which input data to produce the result

- semiring provenance is a mathematical framework for how-provenance

name	dept	
John	1	e_1
Paul	1	e_2
Dave	2	e_3

$\bowtie_{\text{dept}}$

dept	city	
1	N. York	d_1
2	Paris	d_2

name	city	
John	N. York	$e_1 \otimes d_1$
Paul	N. York	$e_2 \otimes d_1$
Dave	Paris	$e_3 \otimes d_2$

Where-provenance

Where provenance – where the result came from, e.g., which input data values were used to produce the result

- centered more on data values than on tuples

name	dept	
John	1	e_1
Paul	1	e_2
Dave	2	e_3

$\bowtie_{\text{dept}}$

dept	city	
1	N. York	d_1
2	Paris	d_2

name	city	
John	N. York	$\{e_1.\text{name}, d_1.\text{city}\}$
Paul	N. York	$\{e_2.\text{name}, d_1.\text{city}\}$
Dave	Paris	$\{e_3.\text{name}, d_2.\text{city}\}$

Provenance

Provenance: additional annotations on data, allowing to propagate useful information on the query results

- how the result has been computed
- how it reacts to changes in the data

Data provenance as additional information: attached to each tuple, computed at query time

Semiring Provenance

Semiring provenance: mathematical foundation of provenance based on annotations being *elements of a semiring*; introduced in the context of relational databases [Green et al., 2007]

Why semirings? – operators verifying axioms

- $\oplus$ corresponds to choices (projections, unions)
- $\otimes$ corresponds to composition (joins, products)

Computational view of provenance – not only informational

Relational Data Model with Annotations

- Relational data model: data decomposed into relations, with labeled attributes. . .
- . . . with an extra provenance annotation for each tuple (think of it first as a tuple id)

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

Relations and Databases

Formally:

- A relational schema $\mathcal{R}$ is a finite sequence of distinct attribute names; the arity of $\mathcal{R}$ is $|\mathcal{R}|$
- A database schema is a mapping from relation names to relational schemas, with finite support
- A tuple over relation schema $\mathcal{R}$ is a mapping from $\mathcal{R}$ to data values; each tuple comes with a provenance annotation
- A relation instance (or relation) over $\mathcal{R}$ is a finite set of tuples over $\mathcal{R}$
- A database instance (or database) over database schema $\mathcal{D}$ is a mapping from the support of $\mathcal{D}$ mapping each relation name R to a relation instance over $\mathcal{D}(R)$

Queries

A query is an arbitrary function that maps databases over a fixed database schema $\mathcal{D}$ to relations over some relational schema $\mathcal{R}$

- In practice, restricted query languages (relational algebra / first order logic, SQL with aggregations)
- The query does not consider or produce any provenance annotations
- . . . we need *semantics* for the provenance annotations of the output, based on that of the input

Semiring Provenance

Semirings

Definition (Semiring)
A semiring is an algebraic structure $(\mathbb{K}, \oplus, \otimes, 0, 1)$ where $\mathbb{K}$ is some set, $\oplus$ and $\otimes$ are binary operators over $\mathbb{K}$, and 0 and 1 are elements of $\mathbb{K}$, satisfying the following axioms:

- $(\mathbb{K}, \oplus, 0)$ is a commutative monoid: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, $a \oplus b = b \oplus a$, $a \oplus 0 = 0 \oplus a = a$;
- $(\mathbb{K}, \otimes, 1)$ is a monoid: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$, $1 \otimes a = a \otimes 1 = a$;
- $\otimes$ distributes over $\oplus$: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$;
- 0 is annihilator for $\otimes$: $0 \otimes a = a \otimes 0 = 0$.

Semirings – Examples

Tropical semiring. $(\mathbb{R}^+ \cup \{\infty\}, \min, +, \infty, 0)$, equivalent to

Viterbi semiring. $([0, 1], \max, \times, 0, 1)$

Top-k semiring. $((\mathbb{R}^+ \cup \{\infty\})^k, \min^k, +^k, (\infty, \ldots, \infty), (0, \infty, \ldots, \infty))$, $k \geq 1$

Counting semiring. $(\mathbb{N} \cup \{\infty\}, +, \times, 0, 1)$

Boolean semiring. $(\{\bot, \top\}, \vee, \wedge, \bot, \top)$

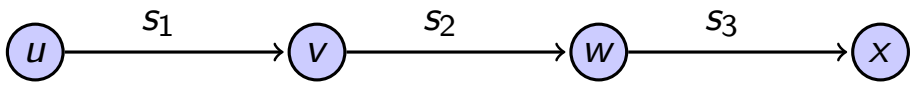
Min-max semiring. $(\mathbb{R}^+ \cup \{\infty\}, \min, \max, \infty, 0)$

k-feature semiring. $((\mathbb{R}^+)^k, \min, \max, (\infty)^k, (0)^k)$, $k \geq 1$

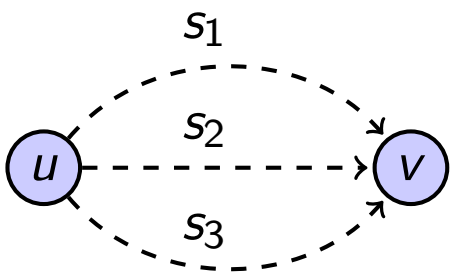
Why semiring. $(2^{\mathcal{T}}, \cup, \cup, \emptyset, \{\epsilon\})$ where $\mathcal{T}$ is a set of tuple ids, and ϵ is the empty set

Integer polynomial semiring. $(\mathbb{N}[X], +, \times, 0, 1)$ where X is a finite set of variables, and $+$, $\times$, 0, 1 are the standard polynomial operators and values

Algebraic Properties



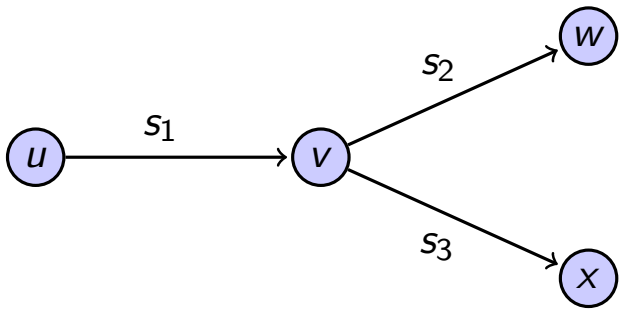
$\otimes$-associativity: $s_1 \otimes s_2 \otimes s_3 := (s_1 \otimes s_2) \otimes s_3 = s_1 \otimes (s_2 \otimes s_3)$



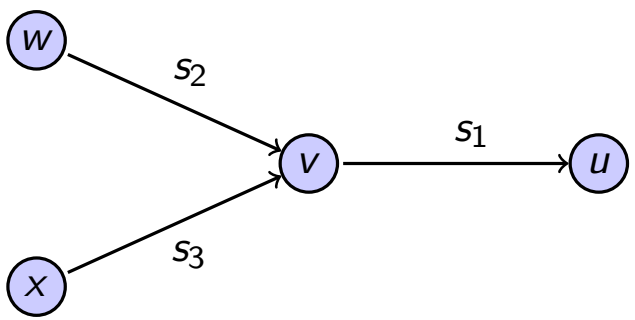
$\oplus$-commutativity: $s_1 \oplus s_2 := s_2 \oplus s_1$

$\oplus$-associativity: $s_1 \oplus s_2 \oplus s_3 := (s_1 \oplus s_2) \oplus s_3 = s_1 \oplus (s_2 \oplus s_3)$

Algebraic Properties



$\otimes$ distributes over $\oplus$: $s_1 \otimes (s_2 \oplus s_3) := (s_1 \otimes s_2) \oplus (s_1 \otimes s_3)$



$\otimes$ distributes over $\oplus$: $(s_1 \oplus s_2) \otimes s_3 := (s_1 \otimes s_2) \oplus (s_1 \otimes s_3)$

Types of Semirings

Commutative: $\forall a, b \in \mathbb{K},\ a \otimes b = b \otimes a$

- e.g.: all examples (but formal languages are not commutative)

Idempotent: $\forall a \in \mathbb{K},\ a \oplus a = a$, natural order $a \sqsubseteq b$ iff $\exists c \in \mathbb{K}$ s.t. $a \oplus c = b$

- e.g.: tropical, Viterbi, Boolean, k-feature

Semiring provenance [Green et al., 2007]

- We fix a semiring $(\mathbb{K}, \oplus, \otimes, 0, 1)$
- We assume provenance annotations are in $\mathbb{K}$
- We consider a query q from the positive relational algebra (selection, projection, renaming, cross product, union)
- We define a semantics for the provenance of a tuple $t \in q(D)$ inductively on the structure of q

Selection, renaming

Provenance annotations of selected tuples are unchanged

$\rho_{\text{name}\to\text{n}}(\sigma_{\text{city}=\text{“New York”}}(R))$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

n	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2

- no transformation is occurring, so the provenance annotations are unchanged

Projection

Provenance annotations of identical, merged, tuples are $\oplus$-ed

$\pi_{\text{city}}(R)$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
New York	$t_1 \oplus t_2$
Paris	$t_3 \oplus t_5 \oplus t_6$
Berlin	$t_4 \oplus t_7$

- only distinct values are kept

Union

Provenance annotations of identical, merged, tuples are $\oplus$-ed

$$\pi_{\text{city}}(\sigma_{\textit{ends-with}(\text{position},\text{“agent”})}(R)) \cup \pi_{\text{city}}(\sigma_{\text{position}=\text{“Analyst”}}(R))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
Paris	$t_3 \oplus t_5$
Berlin	$t_4 \oplus t_7$

- common values are merged as alternatives

Cross product / Joins

Provenance annotations of combined tuples are $\otimes$-ed

$$\pi_{\text{city}}(\sigma_{\textit{ends-with}(\text{position},\text{“agent”})}(R)) \bowtie \pi_{\text{city}}(\sigma_{\text{position}=\text{“Analyst”}}(R))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
Paris	$t_3 \otimes t_5$
Berlin	$t_4 \otimes t_7$

- tuples most exist in both sides to be combined

What can we do with semirings?

counting semiring: count the number of times a tuple can be derived, multiset semantics

Boolean semiring: determines if a tuple exists when a subdatabase is selected

min-max semiring: e.g. security to determine the minimum clearance level required to get a tuple as a result

tropical semiring: minimum-weight way of deriving a tuple (think shortest path in a graph)

Boolean functions: Boolean provenance, as previously defined

integer polynomials: universal provenance, see further

Why-semiring: Why-provenance [Buneman et al., 2001], set of combinations of tuples needed for a tuple to exist

Example of Boolean (function) provenance

$$\pi_{\text{city}}(\sigma_{\text{name}<\text{name2}}(\pi_{\text{name,city}}(R) \bowtie \rho_{\text{name}\to\text{name2}}(\pi_{\text{name,city}}(R))))$$

name	position	city	prov
John	Director	New York	t_1
Paul	Janitor	New York	t_2
Dave	Analyst	Paris	t_3
Ellen	Field agent	Berlin	t_4
Magdalen	Double agent	Paris	t_5
Nancy	HR director	Paris	t_6
Susan	Analyst	Berlin	t_7

city	prov
New York	$t_1 \wedge t_2$
Paris	$(t_3 \wedge t_5) \vee (t_3 \wedge t_6) \vee (t_5 \wedge t_6)$
Berlin	$t_4 \wedge t_7$

Example of counting provenance

$$\pi_{\text{city}}(\sigma_{\text{name}<\text{name2}}(\pi_{\text{name,city}}(R) \bowtie \rho_{\text{name}\to\text{name2}}(\pi_{\text{name,city}}(R))))$$

name	position	city	prov
John	Director	New York	1
Paul	Janitor	New York	1
Dave	Analyst	Paris	1
Ellen	Field agent	Berlin	1
Magdalen	Double agent	Paris	1
Nancy	HR director	Paris	1
Susan	Analyst	Berlin	1

city	prov
New York	1
Paris	3
Berlin	1

Example of security provenance

$$\pi_{\text{city}}(\sigma_{\text{name}<\text{name2}}(\pi_{\text{name,city}}(R) \bowtie \rho_{\text{name}\to\text{name2}}(\pi_{\text{name,city}}(R))))$$

name	position	city	prov
John	Director	New York	unclassified
Paul	Janitor	New York	restricted
Dave	Analyst	Paris	confidential
Ellen	Field agent	Berlin	secret
Magdalen	Double agent	Paris	top secret
Nancy	HR director	Paris	restricted
Susan	Analyst	Berlin	secret

city	prov
New York	restricted
Paris	confidential
Berlin	secret

Provenance polynomials

- Provenance annotations of tuples are polynomials over the provenance annotations of the input tuples
- The provenance annotation of a tuple t is a polynomial describing all the ways t can be derived from the input tuples

Provenance polynomials

$$\pi_{\text{city}}(\sigma_{\text{name}<\text{name2}}(\pi_{\text{name,city}}(R) \bowtie \rho_{\text{name}\to\text{name2}}(\pi_{\text{name,city}}(R))))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

The provenance annotation of the answer *Paris* is the polynomial

$$t_3 t_5 + t_3 t_6 + t_5 t_6$$

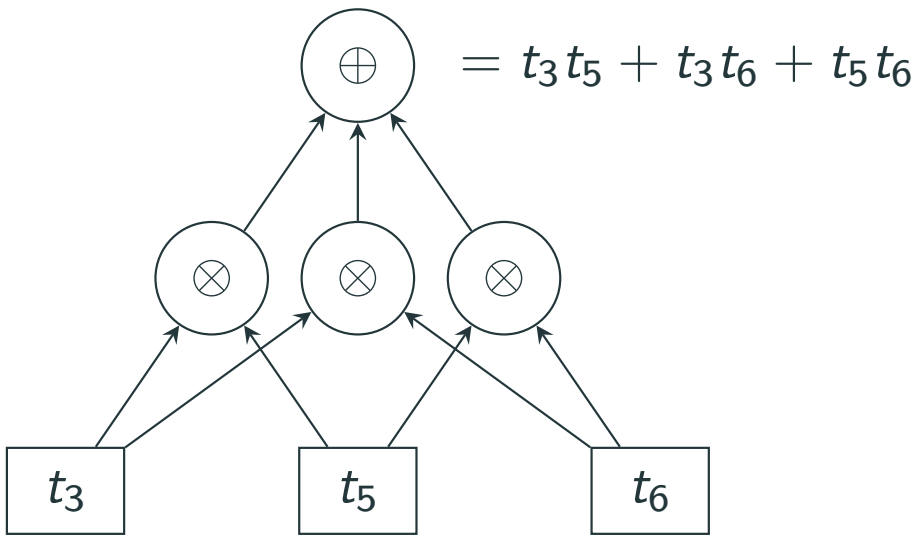
Provenance polynomials [Green et al., 2007]

There is an universal semiring for provenance, the semiring of provenance polynomials $\mathbb{N}[X]$ where X is the set of tuple ids of the input database, and $+$, $\times$, 0, 1 are the standard polynomial operators and values.

- Homomorphisms from $\mathbb{N}[X]$ to any semiring $\mathbb{K}$ give the semantics of $\mathbb{K}$-provenance
- *Example:* the homomorphism mapping all tuple ids to 1 gives the semantics of counting provenance, the homomorphism mapping all tuple ids to $\top$ and all sums to $\vee$ and all products to $\wedge$ gives the semantics of Boolean provenance, etc.

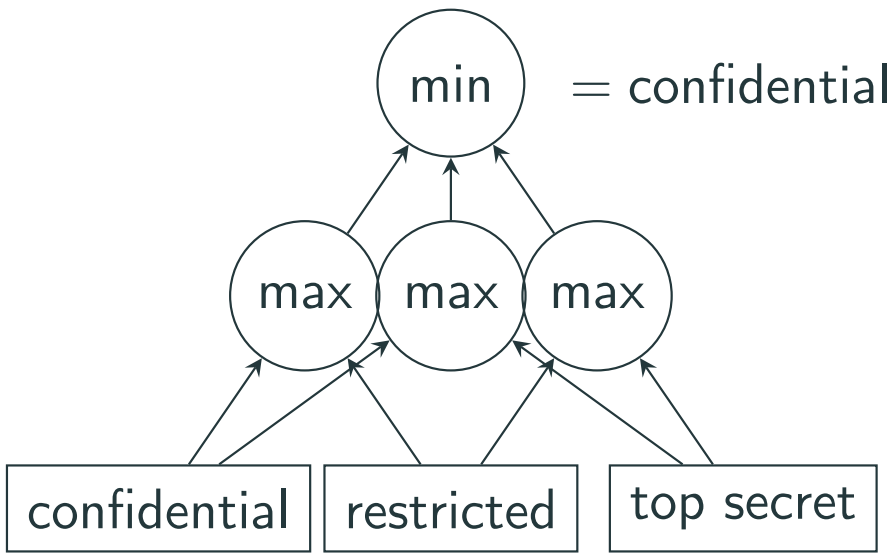
Why do we care about provenance polynomials?

- They allow generic semantics for semiring provenance, via the homorphisms
- They can be represented compactly as provenance circuits



Why do we care about provenance polynomials?

- The provenance for each output tuples can be computed at query time as a provenance circuit in PTIME in the size of the input database
- The provenance circuit can be evaluated in PTIME for any semiring $\mathbb{K}$



Extending to semirings with monus [Amer, 1984, Geerts and Poggi, 2010]

- Some semirings can be equipped with a $\ominus$ verifying:
  - $a \oplus (b \ominus a) = b \oplus (a \ominus b)$
  - $(a \ominus b) \ominus c = a \ominus (b \oplus c)$
  - $a \ominus a = 0 \ominus a = 0$
- Boolean function semiring with $a \wedge \neg b$, Why-semiring with set $\setminus$, counting semiring with truncated difference, ...
- Allows supporting full relational algebra with the $\setminus$ operator, still PTIME

Difference

Provenance annotations of diff-ed tuples are $\ominus$-ed

$$\pi_{\text{city}}(\sigma_{\textit{ends-with}(\text{position},\text{“agent”})}(R)) \setminus \pi_{\text{city}}(\sigma_{\text{position}=\text{“Analyst”}}(R))$$

name	position	city	classification	prov
John	Director	New York	unclassified	t_1
Paul	Janitor	New York	restricted	t_2
Dave	Analyst	Paris	confidential	t_3
Ellen	Field agent	Berlin	secret	t_4
Magdalen	Double agent	Paris	top secret	t_5
Nancy	HR director	Paris	restricted	t_6
Susan	Analyst	Berlin	secret	t_7

city	prov
Paris	$t_5 \ominus t_3$
Berlin	$t_4 \ominus t_7$

Provenance for aggregates [Amsterdamer et al., 2011, Fink et al., 2012]

- Trickier to define provenance for queries with aggregation, even in the Boolean case
- One can construct a K-semimodule $K * M$ for each monoid aggregate M over a provenance database with a semiring in K
- Data values become elements of the semimodule

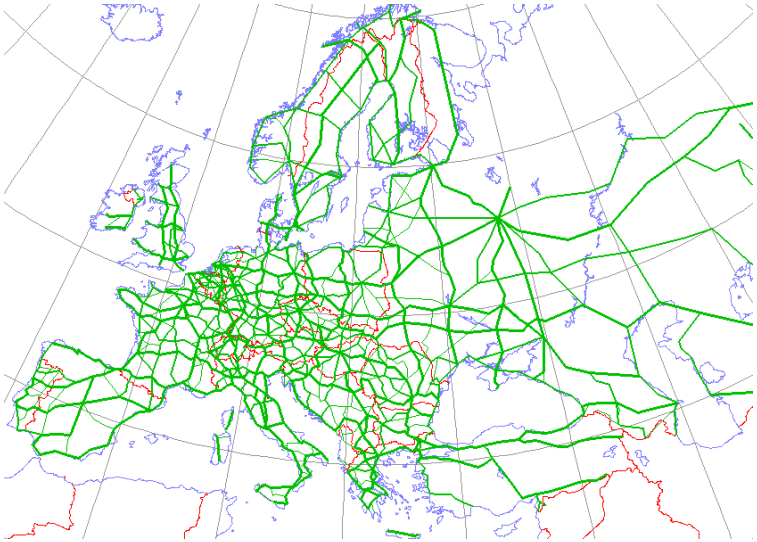
$$\mathrm{count}(\pi_{\mathsf{name}}(\sigma_{\mathsf{city}=\text{“Paris”}}(R)))$$

$$t_3 * 1 + t_5 * 1 + t_6 * 1$$

Graph Semiring Provenance

Queries over graphs – routing in road graphs

Data: graphs are annotated with *distances* (time, physical distance)



Queries: distance between two nodes, nodes being at most at a given distance from a source node

Algorithms: the classic Dijkstra's algorithm, with optimizations for quick answers (contraction hierarchies) – linear in the size of the graph

Queries over graphs – regular pattern queries

Data: graphs are annotated with *labels* (relationships, types of transport links, etc.)

Queries: pairs of nodes which can be reachable via a path having chain of annotations belonging to the language of a given *regular expression*

Algorithms: reachability on the *product graph* between the automaton and the graph – quadratic in the size of the graph

Graph Databases

Graph databases: systems for managing graph-like data

- application in network analysis, transport networks, Semantic Web

Querying on graphs:

- SPARQL, Cypher: usually pattern-matching
- recently, navigational queries (reachability, regular pattern queries) [ISO SC32 / WG3,]

How can we generalize (semiring) provenance to graphs?

Graph Databases with Provenance [Ramusat et al., 2018]

Definition (Graph Database with Provenance)
A graph database with provenance indication (V, E, λ, w) over some semiring $(\mathbb{K}, \oplus, \otimes, 0, 1)$ is an edge-labeled directed graph (V, E, λ) together with a *weight function* $w : E \to \mathbb{K}$.

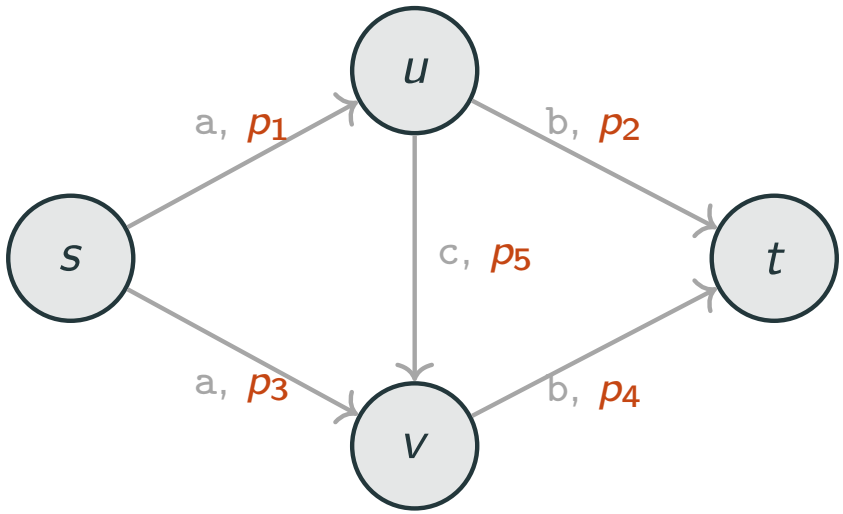
Graph Databases with Provenance [Ramusat et al., 2018]

Definition (Path Provenance)
Let G be a graph database with provenance indication over some semiring $\mathbb{K}$. The provenance between x and y, for x and y two vertices of G is defined as the (possibly infinite) sum:

$$\text{prov}_{\mathbb{K}}(G)(x,y) := w\left[P_{xy}(G)\right] = \bigoplus_{\pi \in P_{xy}(G)} w[\pi].$$

Weight of a path π in G: $w[\pi] := \bigotimes_{i=1}^{l} w[e_i]$

Graph Databases with Provenance – Example



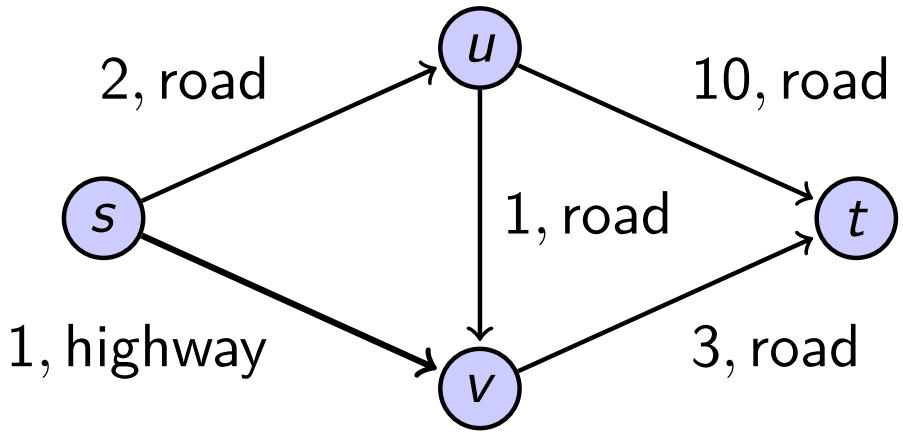
$$\text{prov}_{\mathbb{K}}(G)(s,t) = (p_1 \otimes p_2) \oplus (p_3 \otimes p_4) \oplus (p_1 \otimes p_5 \otimes p_4)$$

Graph Databases with Provenance – Semantics

Semiring provenance in graphs:

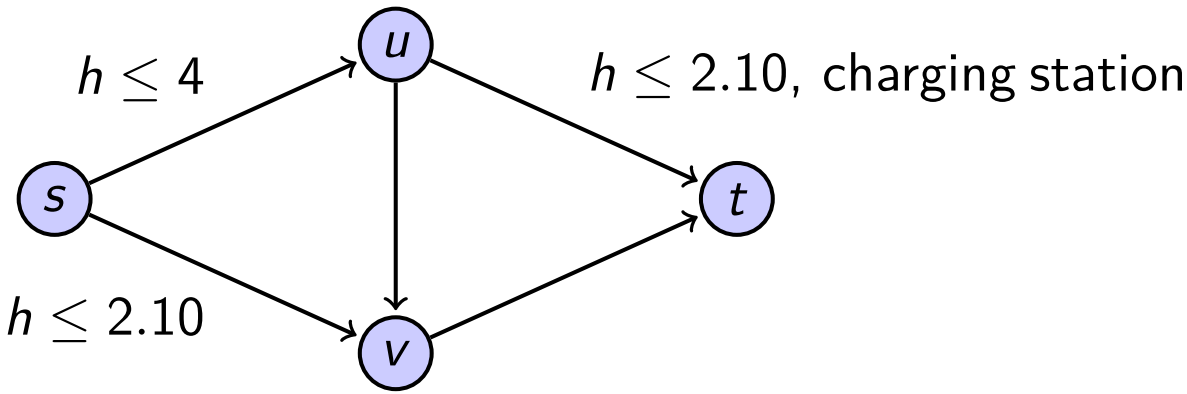
- $\otimes$ encodes an operation aggregating the edges of a given path
- $\oplus$ encodes the alternative between different paths
- provenance between x and y is then taking into account all alternative paths between x and y

Graph Example



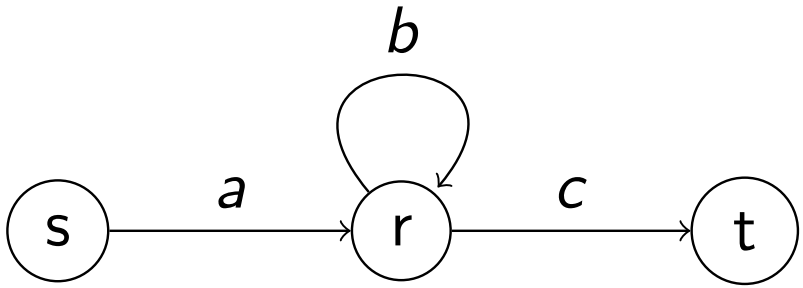
- What is the minimum travel time between s and t? (tropical)
- What is the total number of paths between s and t? (counting)
- What are the best two travel times? (top-k)

Graph Example



- Travel time of 3 meter high vehicles? (k-feature)
- Travel time of vehicles needing charging stations? (k-feature)

Graph Databases with Provenance – Semantics



Provenance of the pair (s, t):

- reason for infinite sum – cycles
- syntactically, star semirings are enough
- interesting conceptually, but not very efficient computationally

Types of Semirings (continued)

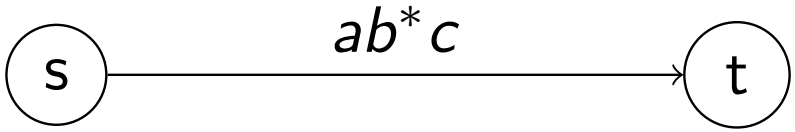
k-closed: $\forall a \in \mathbb{K}$, $\bigoplus_{i=0}^{k+1} a^i = \bigoplus_{i=0}^{k} a^i$, also can define absorptive (0-closed)

- e.g.: tropical, Viterbi, Boolean, k-feature are absorptive; top-k semiring is $(k-1)$-closed

Star semiring: add unary *star operator* * s.t. $a^* = 1 \oplus (a \otimes a^*) = 1 \oplus (a^* \otimes a)$

- e.g.: for absorptive semirings $a^* = 1$, for k-closed $a^* = \bigoplus_{i=0}^{k} a^i$

Graph Databases with Provenance – Semantics



Provenance of the pair (s, t):

- only finite when the star is defined and $a^* = \bigoplus_{i=0}^{\infty} a^i$
- also want $ab^*c = a(\bigoplus_{i=0}^{\infty} a^i)c = \bigoplus_{i=0}^{\infty} ab^i c$
  $\Rightarrow$ c-complete star semirings (or ω-complete...)
- **Still efficient but varies depending on the semiring!**:
  tropical/Vitterbi are linear, top-k is exponential, ...

Graph Databases with Provenance – Semantics of Semirings

Tropical semiring: length of shortest path between s and t.

Viterbi semiring: probability of most likely path between s and t.

Top-k semiring: lengths of k shortest paths between s and t.

Counting semiring: total number of paths between s and t, edge weights being interpreted as number of edges between two vertices.

Boolean semiring: existence of a path between s and t, depending on the existence of edges denoted by their Boolean weights.

k-feature semiring: minimum feature value along each dimension of all paths between s and t; if min and max are exchanged, maximum feature value along some path from s to t.

Implementing Provenance in DBMS

Provenance in DBMS

Many systems for provenance in databases, e.g.
Perm [Glavic and Alonso, 2009], GProM [Arab et al., 2018], ...

- most recent is ProvSQL [Sen et al., 2026], a PostgreSQL extension for semiring provenance

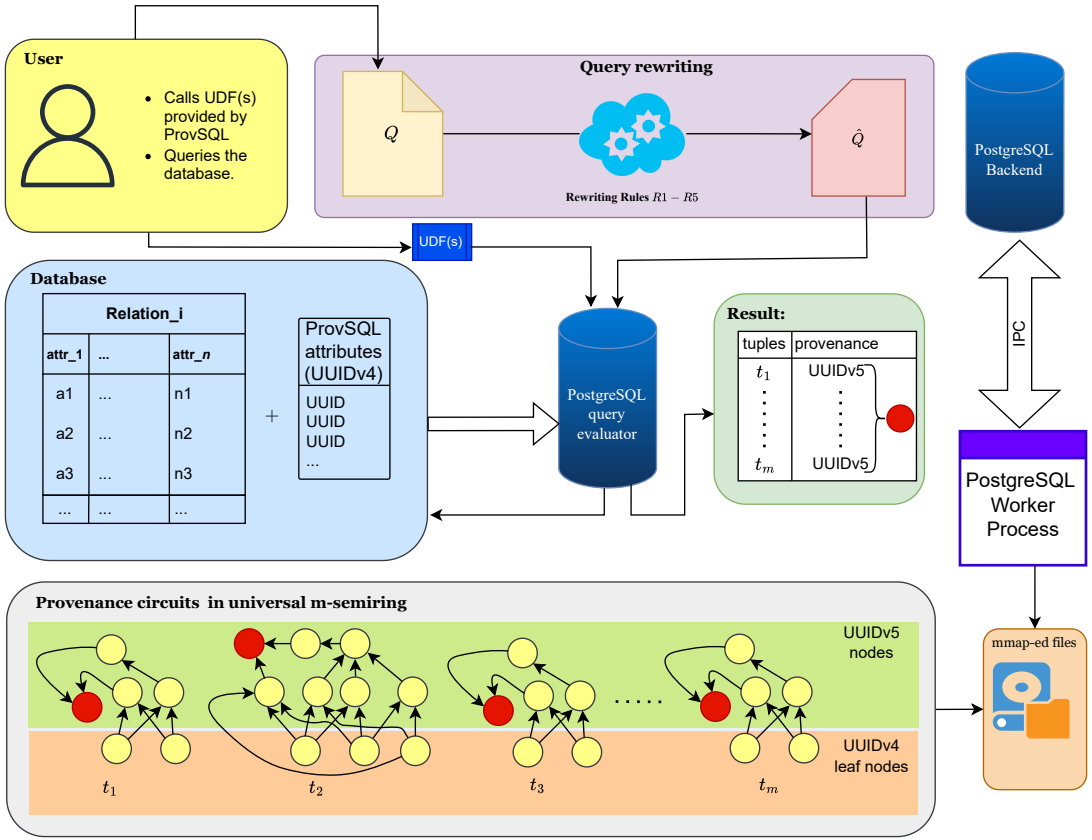
ProvSQL:

- supports any semiring provenance, including aggregates and difference
- efficient algorithms for provenance computation via query rewriting and provenance circuits
- open-source and available at `https://provsql.org/` – try it, tutorial available!

ProvSQL: Annotated Tables

Generic annotations for tuples, e.g. t_1, t_2, ... via a `prov` column and UUIDs for tuples

Generic provenance computation via query rewriting and keeping circuits representations of provenance in the `prov` column of query results



ProvSQL: Query Rewriting

From

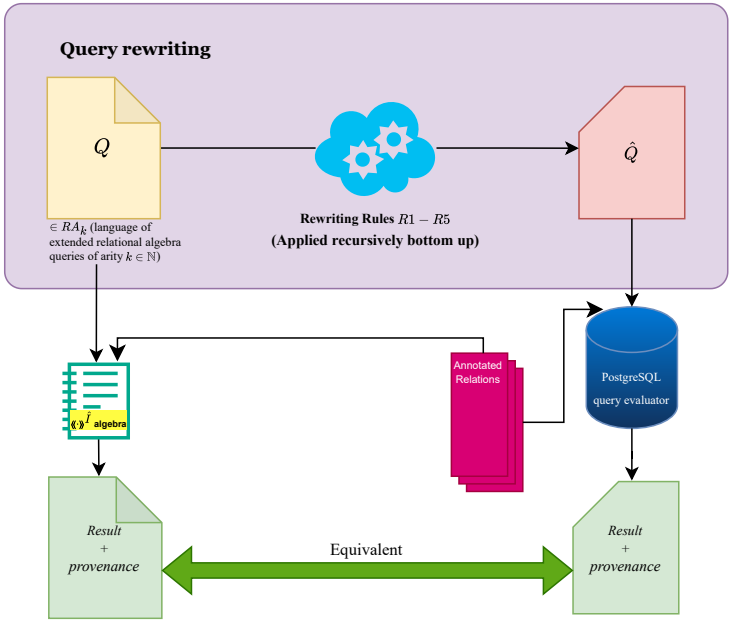
```
SELECT DISTINCT p1.city
 FROM personnel p1 JOIN personnel p2
 ON p1.city = p2.city AND p1.id < p2.id
```

to

```
SELECT DISTINCT p1.city, PROVENANCE() as prov
 FROM personnel p1 JOIN personnel p2
  ON p1.city = p2.city AND p1.id < p2.id
  GROUP BY p1.city
```

Rewrite the query as an aggregation via a `PROVENANCE` UDF:

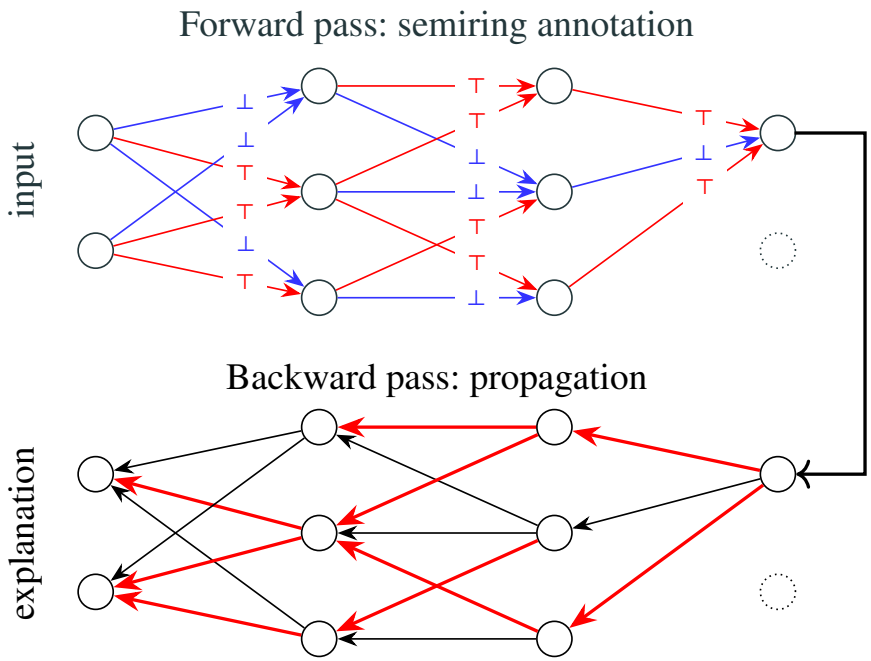
- each tuple in the original query $\rightarrow$ the provenance is the aggregation of the semiring rules for that tuple



Using Provenance for Explainability

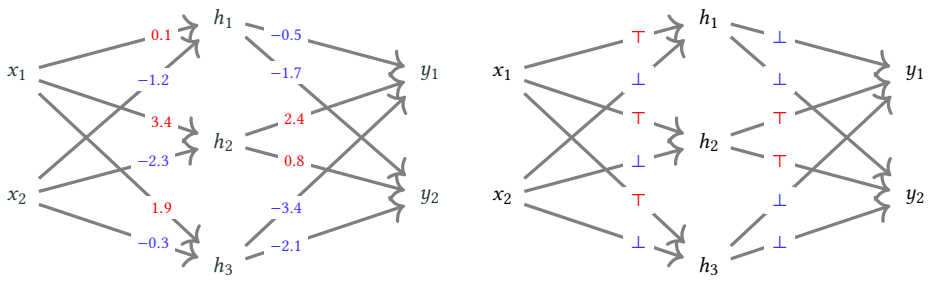
LRP with Semiring Provenance [Groudiev et al., 2025]

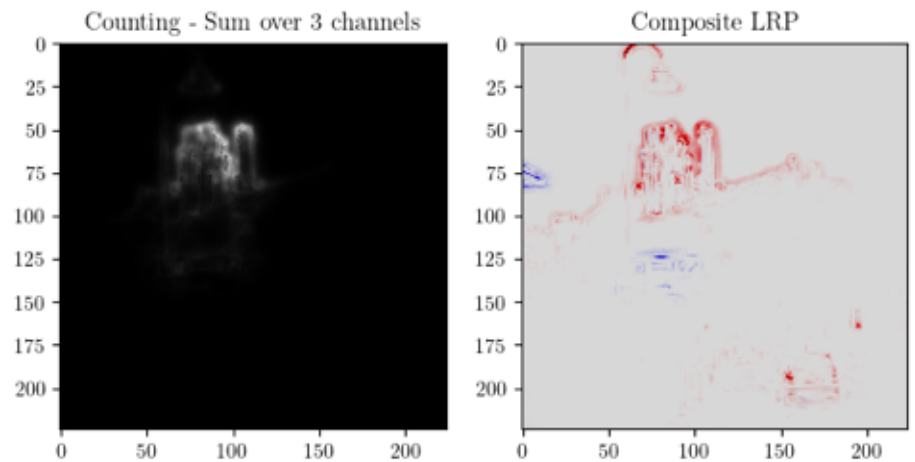
- LRP (Layer-wise Relevance Propagation) [Bach et al., 2015]: explain a neural network by propagating output relevance backward to input features using weighted contributions
- Extension: replace real arithmetic with semiring operations $(\mathbb{K}, \oplus, \otimes)$
  - **Forward pass:** annotate each neuron with a semiring element
  - **Backward pass:** propagate relevance via $\oplus$ (aggregation) and $\otimes$ (weighting)



LRP with Semiring Provenance [Groudiev et al., 2025]

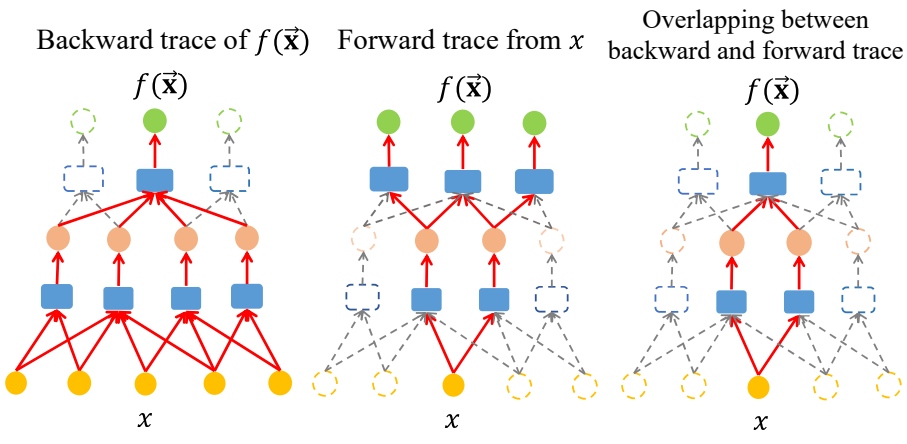
- Same semirings as for graph provenance → different explanation granularities: Boolean (relevant regions), counting (number of relevant paths), Viterbi (highest-confidence path)
- **Link to provenance:** a neural network's forward pass is a circuit





Provenance Circuits for Explainability [Zhang et al., 2024]

- PXAI: record model inference as a provenance DAG tracking creation and transformation of every variable
- **Backward trace** from output → subgraph of relevant computations; **forward trace** from input → what it affects
- **Link to provenance:** the provenance circuit and a neural network are the same structure; intersecting forward and backward traces gives a smaller circuit



Take Aways and Open Issues

- Practical provenance for (graph) database systems is possible, with rich semantics and efficient algorithms
- **Is provenance good for explainability in ML?**
  - provenance circuits can be quite large in NNs $\rightarrow$ need for summarization, abstraction, etc.
  - how to evaluate the quality of provenance-based explanations?
  - how to present provenance-based explanations to users $\rightarrow$ provenance to NL, visualizations, etc.

References i


Amer, K. (1984).
Equationally complete classes of commutative monoids with monus.
Algebra Universalis, 18(1).


Amsterdamer, Y., Deutch, D., and Tannen, V. (2011).
Provenance for aggregate queries.
In *PODS*.


Arab, B. S., Feng, S., Glavic, B., Lee, S., Niu, X., and Zeng, Q. (2018).
GProM - A swiss army knife for your provenance needs.
IEEE Data Eng. Bull., 41(1):51–62.

References ii

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015).
On pixel-wise explanations for Non-Linear classifier decisions by layer-wise relevance propagation.
PLOS ONE, 10(7):e0130140.

Buneman, P., Khanna, S., and Tan, W. C. (2001).
Why and where: A characterization of data provenance.
In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*

References iii


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


Fink, R., Han, L., and Olteanu, D. (2012).
Aggregation in probabilistic databases via knowledge compilation.
Proceedings of the VLDB Endowment, 5(5):490–501.


Geerts, F. and Poggi, A. (2010).
On database query languages for k-relations.
J. Applied Logic, 8(2).

References iv


Glavic, B. and Alonso, G. (2009).
Perm: Processing provenance and data on the same data model through query rewriting.
In *ICDE*, pages 174–185.


Green, T. J., Karvounarakis, G., and Tannen, V. (2007).
Provenance semirings.
In *PODS*, pages 31–40.


Groudiev, A., Saha, A., and Maniu, S. (2025).
Extending layer-wise relevance propagation in neural networks using semiring annotations.
In *ProvenanceWeek*.

References

ISO SC32 / WG3.
Graph Query Language GQL.
https://www.gqlstandards.org/.

Ramusat, Y., Maniu, S., and Senellart, P. (2018).
Semiring provenance over graph databases.
In *TaPP*.

Sen, A., Maniu, S., and Senellart, P. (2026).
ProvSQL: A General System for Keeping Track of the Provenance and Probability of Data.
In *Proc. ICDE*, Montréal, Canada.
Also CoRR abs/2504.12058.

W3C (2024).
Prov overview.


Zhang, J., Zhou, W., and Ujcich, B. E. (2024).
Provenance-Enabled Explainable AI.
Proc. ACM Manag. Data, 2(6):250:1–250:27.