CS 595 - Hot topics in database systems:

Data Provenance

I. Database Provenance
I.1 Provenance Models and Systems

Boris Glavic

September 24, 2012
How-Provenance, Semirings, and Orchestra

Introduction
Semiring Semantics for Relational Algebra
How-Provenance or Provenance Polynomials
Relationship to other Provenance Models
ORCHESTRA
Recap
How-Provenance

Introduction

How-Provenance

Rationale

- In addition to model which tuples influenced a tuple
- ...model how tuples where combined in the computation
  - Alternative use: need one of the tuples (e.g., union)
  - Conjunctive use: need all tuples together (e.g., join)

Representation

- Formulas over operators and variables
  - Operators define how tuples where combined
  - Variables represent tuples (one variable per tuple)
### Approach

**Alternative Semantics for the Relational Model**
- Tuples are annotated with elements from a semiring
- Define relational algebra operators using the operators of the semiring
- Prove it coincides with set- or bag-semantics for certain semirings

**How-Provenance**
- Use special semiring that generalizes all semirings
- Elements are symbolic computations
**Approach**

**ORCHESTRA**

- Collaborative Data Sharing System
- Independent peers with their own database schema and instance
- Schema mappings between peers schemata
- Peers periodically exchange updates
- Provenance to compute trust in update and deletion propagation
Excursion: Semirings

Commutative Monoids

- \((K, +, 0)\)
- A set \(K\)
- An operation \(K \rightarrow K\) (say +) with neutral element 0:
  - \((a + b) + c = a + (b + c)\) (associativity)
  - \(0 + a = 0 + a = a\) (neutral element)
  - \(a + b = b + a\) (associativity)

Example

- \((\mathbb{N}, +, 0)\) - Natural numbers addition
- \((\mathbb{N}, \times, 1)\) - Natural numbers multiplication
- \((\mathbb{B}, \land, true)\): \(\mathbb{B} = \{true, false\}\) - Conjunction over boolean constants
- \((\mathbb{B}, \lor, false)\) - Disjunction over boolean constants
Excursion: Semirings

Commutative Semiring

- \((K, +, \times, 0, 1)\)
- Set \(K\) with operations \(+\) and \(\times\) (neutral elements 0 and 1)
- \((K, +, 0)\) and \((K, \times, 1)\) are commutative monoids
- \(a \times (b + c) = (a \times b) + (a \times c)\) (Distributivity)
- \((a + b) \times c = (a \times c) + (b \times c)\) (Distributivity)
- \(a \times 0 = 0 \times a = 0\) (multiplication with 0)

Example

- \((\mathbb{N}, +, \times, 0, 1)\) - Natural numbers with addition and multiplication
- \((\mathbb{B}, \lor, \land, \text{false}, \text{true})\) - Conjunctions and disjunctions over boolean constants
Homomorphism

**Definition**

Homomorphism

- Given two semirings $K$ and $K'$
- A function from $K$ to $K'$ is a homomorphism $h$ iff:
  - $h(a + b) = h(a) + h(b)$
  - $h(a \times b) = h(a) \times h(b)$
  - $h(0) = 0$
  - $h(1) = 1$

- Homomorphism from $K$ to $K' \Rightarrow K$ is more general than $K'$
Free Objects

- Given an algebraic structure a free object is one with an homomorphism into all other objects of this type
  - E.g., the free commutative semiring is an structure with homomorphism into all other commutative semirings
Free Objects

• Given an algebraic structure a free object is one with an homomorphism into all other objects of this type
  • E.g., the free commutative semiring is an structure with homomorphism into all other commutative semirings

⇒ The free semiring is the most general semiring
Free Objects

- Given an algebraic structure a free object is one with an homomorphism into all other objects of this type
  - E.g., the free commutative semiring is an structure with homomorphism into all other commutative semirings

- The **free semiring** is the most general semiring

- Only equivalences enforced by the structure being semiring can hold
  - For any additional equivalence: Find semiring where equivalence does not hold \(\Rightarrow\) No homomorphism! contradiction
Free Objects

- Given an algebraic structure a free object is one with an homomorphism into all other objects of this type
  - E.g., the free commutative semiring is a structure with homomorphism into all other commutative semirings
  - \( \Rightarrow \) The free semiring is the most general semiring
  - \( \Rightarrow \) Only equivalences enforced by the structure being semiring can hold
    - For any additional equivalence: Find semiring where equivalence does not hold \( \Rightarrow \) No homomorphism! contradiction
  - \( \Rightarrow \) Elements of free semiring are uninterpreted expressions
    - Placeholders for semiring elements
    - Do not interpret semiring operation
Free Objects

Example

- \((a + b) \times c\) is an element
- \(k_1 = (a + b)\) and \(k_2 = (c \times d)\): \(k_1 + k_2 = (a + b) + (c \times d)\)
Outline

1. How-Provenance, Semirings, and Orchestra
   - Introduction
   - Semiring Semantics for Relational Algebra
   - How-Provenance or Provenance Polynomials
   - Relationship to other Provenance Models
   - ORCHESTRA
   - Recap
K-Relations
- **U-tuple**: tuples over set of attributes $U$
  - $U - \text{Tup} = \text{set of all } U\text{-tuples}$
- Semiring $K$
- A *K*-relation $R$ over a set of attributes $U$ is
  - function $U - \text{Tup} \rightarrow K$
  - $\text{support}(R) = \{t \mid R(t) \neq 0\}$ is finite

Notation
- $K = \mathbb{N}$
Interpretations of Semirings

Semiring Interpretations

- \((\mathbb{N}, +, \times, 0, 1)\): Tuples annotated with integers \(\Rightarrow\) Bag-semantics

Example

<table>
<thead>
<tr>
<th>R</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
How-Provenance

Semiring Semantics for Relational Algebra

Interpretations of Semirings

Semiring Interpretations

- \((\mathbb{B}, \lor, \land, \text{false}, \text{true})\): \(\mathbb{B} = \{\text{false}, \text{true}\}\): Tuples with true/false annotations \(\Rightarrow\) Set-semantics

Example

<table>
<thead>
<tr>
<th></th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>true</td>
<td>2</td>
</tr>
</tbody>
</table>
Semiring Interpretations

- \((\text{PosBool}(X), \lor, \land, \text{false}, \text{true})\): \text{PosBool}(X) = \text{set of variables: Tuples annotated with boolean expressions} \Rightarrow \text{c-tables (probabilistic databases)}

Example

\[
\begin{array}{c|c|c}
\text{R} & \text{a} & \\
\hline
x_1 \lor (x_2 \land x_3) & 1 & \\
\hline
x_4 & 2 & \\
\end{array}
\]
Relational Algebra for $K$-relations

Rationale

- Express relational algebra operators as semiring operations
- **Sanity checks:**
  - For $K = \mathbb{B} \Rightarrow$ same results (equivalences) as set-semantics
  - For $K = \mathbb{N} \Rightarrow$ same results (equivalences) as bag-semantics
Operator Definitions

**Selection**

- \((\sigma_C(R))(t) = R(t) \times C(t)\)
- Selection predicate \(C\) is function \(U - \text{Tup} \rightarrow \{0, 1\}\)
  - Recall \(a \times 0 = 0\) and \(a \times 1 = a\)

**Projection**

- \((\pi_A(R))(t) = \sum_{t = t'.A} R(t')\)
- \(A \subseteq U\)

**Union**

- \((R_1 \cup R_2)(t) = R_1(t) + R_2(t)\)
Operator Definitions

**Natural Join**

\[ (R_1 \Join R_2)(t) = R_1(t_1) \times R_2(t_2) \]

- \( t_1 = t.U_1 \)
- \( t_2 = t.U_2 \)

**Renaming**

\[ (\rho_\beta(R))(t) = R(t \circ \beta) \]

- \( \beta : U \rightarrow U' \) attribute renaming
How-Provenance
Semiring Semantics for Relational Algebra

Evaluation Example

- Semiring is $\mathbb{N}$
- $q = \sigma_{a=1}(\pi_a(R))$
- $q(t) = \sum_{t'.a=t} R(t') \times (a = 1)(t)$

\[
\begin{array}{c|cc}
 a & b \\
\hline
 2 & 1 & 3 \\
 3 & 1 & 4 \\
 2 & 2 & 4 \\
\end{array}
\]

\[
\begin{array}{c}
 Q \\
\hline
 a \\
\hline
 1 \\
 2 \\
\end{array}
\]

\[
(2 + 3) \times 1 = 5 \\
2 \times 0 = 0
\]
Equivalence Examples

- **Union**:
  - Associative: \( R \cup (S \cup T) = (R \cup S) \cup T \)
  - Commutative: \( R \cup S = S \cup R \)
  - Identity \( \emptyset \): \( R \cup \emptyset = R \)

- **Join**
  - Associative: \( R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T \)
  - Commutative: \( R \bowtie S = S \bowtie R \)

- **Selection**
  - \( \sigma_{false}(R) = \emptyset \)
  - \( \sigma_{true}(R) = R \)
Homomorphisms in Query Evaluation

Homomorphisms commute with Query Evaluation

- $Q(h(I)) = h(Q(I))$
- $\Rightarrow$ We can apply $h$ either before or after evaluating the query without affecting the result

Example

- Homomorphism from $\mathbb{N}$ (bag-semantics) to $\mathbb{B}$ (set-semantics):
  $h(n) = true$ except $h(0) = false$

- E.g., $\sigma_{a>1}(R)$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>true</td>
<td>2</td>
</tr>
</tbody>
</table>
Outline

1. How-Provenance, Semirings, and Orchestra
   - Introduction
   - Semiring Semantics for Relational Algebra
   - How-Provenance or Provenance Polynomials
   - Relationship to other Provenance Models
   - ORCHESTRA
   - Recap
Provenance Polynomials

Rationale

- Use semiring annotations to model provenance
- Annotate a query result tuple with the semiring expression that was used to compute it
- ⇒ need free semiring

Provenance Polynomials Semiring

- \((\mathbb{N}[I], +, \times, 0, 1)\)
- \(\mathbb{N}[I]\) - Polynomials with natural number exponents
  - Variables: One per tuple in \(I\)
- Convention: annotate each instance tuple with a variable named after its tuple \(id\)
How-Provenance or Provenance Polynomials

Provenance Polynomials Example

$q = \pi_a(R)$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>t₂</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

$Q$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁ + t₂</td>
<td>1</td>
</tr>
</tbody>
</table>
How-Provenance or Provenance Polynomials

Provenance Polynomials Example II

Example

\[ q = \pi_{\text{Name}}(E \Join \sigma_{\text{Dep}=\text{CS}}(P) \Join A) \]

<table>
<thead>
<tr>
<th>Employee</th>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>1</td>
<td>Peter</td>
</tr>
<tr>
<td>e₂</td>
<td>2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>e₂</td>
<td>3</td>
<td>Michael</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assigned</th>
<th>PName</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>Server</td>
<td>1</td>
</tr>
<tr>
<td>a₂</td>
<td>Server</td>
<td>2</td>
</tr>
<tr>
<td>a₃</td>
<td>Webpage</td>
<td>2</td>
</tr>
<tr>
<td>a₄</td>
<td>Fire CS</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project</th>
<th>PName</th>
<th>Dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁</td>
<td>Server</td>
<td>CS</td>
</tr>
<tr>
<td>p₂</td>
<td>Webpage</td>
<td>CS</td>
</tr>
<tr>
<td>p₃</td>
<td>Fire CS</td>
<td>HR</td>
</tr>
</tbody>
</table>
Provenance Polynomials Example II

Example

\[ q = \pi_{Name}(E \bowtie \sigma_{Dep=CS}(P) \bowtie A) \]
\[ (q)(t) = \sum_{u.A=t} E(u.E) \times P(u.P) \times (Dep = CS)(u.P) \times A(u.P) \]

\[ e_1 \times a_1 \times p_1 \]
\[ (e_2 \times a_2 \times p_1) + (e_2 \times a_3 \times p_2) \]

<table>
<thead>
<tr>
<th>Employee</th>
<th>Assigned</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td><strong>PName</strong></td>
<td><strong>Dep</strong></td>
</tr>
<tr>
<td>e_1</td>
<td>Server</td>
<td>CS</td>
</tr>
<tr>
<td>e_2</td>
<td>Server</td>
<td>CS</td>
</tr>
<tr>
<td>e_3</td>
<td>Webpage</td>
<td>CS</td>
</tr>
<tr>
<td>e_4</td>
<td>Fire CS</td>
<td>HR</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>1</td>
</tr>
<tr>
<td>Gertrud</td>
<td>2</td>
</tr>
<tr>
<td>Michael</td>
<td>3</td>
</tr>
</tbody>
</table>
The Fundamental Property

- The semiring of provenance polynomials is the free commutative semiring
- \( \Rightarrow \) there exists a homomorphism from \( \mathbb{N}[I] \) into any commutative semiring
- \( \text{Eval}_K : \mathbb{N}[I] \rightarrow K \) is this unique homomorphism defined as
  - Replace each tuple variable \( t \) with the element of \( K \) assigned to the tuple represented by \( t \)
  - Interpret the abstract operations from \( \mathbb{N}[I] \) as operations from \( K \)
Example Application of the Fundamental Property

\[ q = \pi_a(R) \]

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>t_2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ t_1 + t_2 \]

\[ a \]

1
Example Application of the Fundamental Property

\[ q = \pi_a(R) \]

Interpretation in \( \mathbb{N} \)

<table>
<thead>
<tr>
<th>R</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

\[ t_1 + t_2 \]
Example Application of the Fundamental Property

\[ q = \pi_a(R) \]

Interpretation in \( \mathbb{N} \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ 2 + 1 = 3 \]

Q

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
The “How” Part

Interpretation of + and ×

- **+: Alternative use of tuples**
  - Operators: Union, Projection
  - Check set-semantics: only one tuples is need \( \Rightarrow \lor \) as + operation
  - Check bag-semantics: multiplicities are additive \( \Rightarrow \) natural number addition as +

- **×**: Conjunctive use of tuples
  - Operations: Join
  - Check set-semantics: both tuples are needed \( \Rightarrow \land \) as × operation
  - Check bag-semantics: multiplicities of matching tuples are multiplied \( \Rightarrow \) natural number multiplication as ×
How-Provenance or Provenance Polynomials

Insensitivity to Query Rewrite

Bag-semantics

- Modelling relational algebra as commutative semiring operations
  - Possible, because same equivalences
- $\mathbb{N}[/]$ is free commutative semiring
- $\Rightarrow$ Equivalences for $\mathbb{N}[/]$ and bag-semantics are the same!
- $\Rightarrow$ $\mathbb{N}[/]$ is insensitive
How-Provenance or Provenance Polynomials

Insensitivity to Query Rewrite

Set-semantics

- \( \mathbb{N}[I] \) no longer insensitive
  - E.g., \( R \not\equiv R \bowtie R \)
- \( \mathbb{B}[I] \): polynomials with boolean coefficients and exponents has same equivalences as set semantics
- There exists an homomorphism from \( \mathbb{N}[I] \) to \( \mathbb{B}[I] \) (\( \mathbb{N}[I] \) is free object!)
- \( \Rightarrow \) apply equivalences of \( \mathbb{B}[I] \) to \( \mathbb{N}[I] \) then insensitive for set-semantics
How-Provenance or Provenance Polynomials

How-provenance

Notation

- We write $\mathbb{N}[l](q, t)$ for $(q)(t)$ evaluated in $\mathbb{N}[l]$
- also use this for other semirings $K$
How-Provenance or Provenance Polynomials

Beyond Positive Relational Algebra

**Set Difference**
- Need additional operator $-$
- $\Rightarrow$ from semiring to structures $(S, +, \times, -, 1, 0)$
  - Different equivalences hold!
- Provenance use (more complex) free object for such structures

**Aggregation**
- Annotate attribute values with combinations of
  - tuple semiring provenance
  - annotation for values for computations on values (representing aggregation)
Outline

1. How-Provenance, Semirings, and Orchestra
   - Introduction
   - Semiring Semantics for Relational Algebra
   - How-Provenance or Provenance Polynomials
   - Relationship to other Provenance Models
   - ORCHESTRA
   - Recap
Relationship of Provenance Polynomials and other Provenance Models

Rationale

- How is the provenance polynomials model related to other provenance models?
- Can we find semirings that models, e.g., Why-Provenance?
Why-Provenance

Semiring

- $K_{\text{Why}} = (\mathcal{P}(\mathcal{P}(I)), \cup, \uplus, \emptyset, \{\emptyset\})$
- $\mathcal{P} = \text{powerset}$
- $\Rightarrow \mathcal{P}(\mathcal{P}(I))$ is all sets containing subsets of the instance $I$
- $\Rightarrow$ all potential sets of witnesses
- $\uplus$ is normal set union
- $\times$ is $S_1 \uplus S_2 = \{(a \cup b) \mid a \in S_1 \land b \in S_2\}$
  - $\Rightarrow$ pairwise union
  - $\Rightarrow$ combining witnesses
### Why-Provenance

#### Example

\[
q = \pi_{\text{Name}}(E \Join \sigma_{\text{Dep}=CS(P)} \Join A) \\
(q)(t) = \sum_{u.A=t} E(u.E) \times P(u.P) \times (\text{Dep} = CS)(u.P) \times A(u.P)
\]

<table>
<thead>
<tr>
<th>Employee</th>
<th>Assigned</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td><strong>Name</strong></td>
<td><strong>PName</strong></td>
</tr>
<tr>
<td>{{e_1}}</td>
<td>1</td>
<td>Peter</td>
</tr>
<tr>
<td>{{e_2}}</td>
<td>2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>{{e_2}}</td>
<td>3</td>
<td>Michael</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Employee: Peter, Gertrud, Michael
- Assigned: Server, Webpage, Fire CS
- Project: Server, Webpage, Fire CS
Insensitive Why-Provenance

**Semiring**

- \( K_{\text{Why}} = (\min(\mathcal{P}(\mathcal{P}(I))), \cup_{\text{min}}, \cup_{\text{min}}, \emptyset, \{\emptyset\}) \)
- \( \Rightarrow \mathcal{P}(\mathcal{P}(I)) \) is all sets containing subsets of the instance \( I \)
- \( \min(S) = \{a \mid a \in S \land \forall b \in S : b \subseteq a\} \)
- \( S_1 \cup_{\text{min}} S_2 = \min(S_1 \cup S_2) \)
- \( S_1 \cup_{\text{min}} S_2 = \min(S_1 \uplus S_2) \)
- \( \Rightarrow \text{Same operations, compute minimal elements} \)
How-Provenance

Relationship to other Provenance Models

Insensitive Why-Provenance

Example

\[ q = \pi_{\text{Name}}(E \bowtie \sigma_{\text{Dep}=\text{CS}}(P) \bowtie A) \]
\[ (q)(t) = \sum_{u.A=t} E(u.E) \times P(u.P) \times (\text{Dep} = \text{CS})(u.P) \times A(u.P) \]

Q

<table>
<thead>
<tr>
<th>Name</th>
<th>Employee</th>
<th>Assigned</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{{e_1}}</td>
<td>{{a_1}}</td>
<td>{{p_1}}</td>
</tr>
<tr>
<td></td>
<td>1 Peter</td>
<td>1 Server</td>
<td>Server</td>
</tr>
<tr>
<td></td>
<td>2 Gertrud</td>
<td>2 Server</td>
<td>Webpage</td>
</tr>
<tr>
<td></td>
<td>3 Michael</td>
<td>3 Webpage</td>
<td>Fire CS</td>
</tr>
</tbody>
</table>

Slide 24 of 36 Boris Glavic CS 595 - Hot topics in database systems: Data Provenance
Relationship to other Provenance Models

**Insensitive Why-Provenance**

**Semiring**

- $K_{IWHy} = (\min(\mathcal{P}(\mathcal{P}(I))), \cup_{\min}, \cup_{\min}, \emptyset, \{\emptyset\})$
- $\Rightarrow \mathcal{P}(\mathcal{P}(I))$ is all sets containing subsets of the instance $I$
- $\min(S) = \{a \mid a \in S \land \forall b \in S : b \subseteq a\}$
- $S_1 \cup_{\min} S_2 = \min(S_1 \cup S_2)$
- $S_1 \cup_{\min} S_2 = \min(S_1 \cup S_2)$
- $\Rightarrow$ Same operations, compute minimal elements
### Insensitive Why-Provenance

**Example**

\[ q = \pi_{\text{Name}}(E \bowtie \sigma_{\text{Dep}=\text{CS}}(P) \bowtie A) \]

\[ (q)(t) = \sum_{u.A=t} E(u.E) \times P(u.P) \times (\text{Dep} = \text{CS})(u.P) \times A(u.P) \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Employee</th>
<th>Assigned</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{{e_1}, {e_2}}</td>
<td>{{e_1, a_1, p_1}, {e_2, a_2, p_1}, {e_2, a_3, p_2}}</td>
<td>{{p_1}, {p_2}, {p_3}}</td>
</tr>
<tr>
<td>Peter</td>
<td>1 Peter</td>
<td>{a_1}</td>
<td>Server 1 CS</td>
</tr>
<tr>
<td>Gertrud</td>
<td>2 Gertrud</td>
<td>{a_2}</td>
<td>Server 2 CS</td>
</tr>
<tr>
<td></td>
<td>3 Michael</td>
<td>{a_3}</td>
<td>Webpage 2 CS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{a_4}</td>
<td>Fire CS 3 HR</td>
</tr>
</tbody>
</table>
Lineage

Different Model

- The inventors of provenance polynomials consider a slightly different Lineage model
- Provenance is a set of tuples instead of a list of sets of tuples

Semiring

- $K_{Lin} = (\mathcal{P}(I), \cup_\perp, \cup^*_\perp, \perp, \emptyset)$
- $\Rightarrow \mathcal{P}(I)$ is all subsets of the instance $I$
- $\perp$ is a not defined element
- $\cup_\perp$ and $\cup^*_\perp$ are union with different behaviour on $\perp$
- $\perp \cup_\perp S = S \cup_\perp \perp = S$
- $\perp \cup^*_\perp S = S \cup^*_\perp \perp = \emptyset$
Relationship to other Provenance Models

Lineage

Example

\[ q = \pi_{Name}(E \bowtie \sigma_{Dep=CS}(P) \bowtie A) \]

\[ (q)(t) = \sum_{u.A=t} E(u.E) \times P(u.P) \times (Dep = CS)(u.P) \times A(u.P) \]

\{ e_1, a_1, p_1 \}

\{ e_2, a_2, a_3, p_1, p_2 \}

Employee

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
</tr>
<tr>
<td>2</td>
<td>Gertrud</td>
</tr>
<tr>
<td>3</td>
<td>Michael</td>
</tr>
</tbody>
</table>

Assigned

<table>
<thead>
<tr>
<th>PName</th>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>1</td>
</tr>
<tr>
<td>Server</td>
<td>2</td>
</tr>
<tr>
<td>Webpage</td>
<td>2</td>
</tr>
<tr>
<td>Fire CS</td>
<td>3</td>
</tr>
</tbody>
</table>

Project

<table>
<thead>
<tr>
<th>PName</th>
<th>Dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>CS</td>
</tr>
<tr>
<td>Webpage</td>
<td>CS</td>
</tr>
<tr>
<td>Fire CS</td>
<td>HR</td>
</tr>
</tbody>
</table>
"Real" Lineage

Can we also model the list of sets of tuples lineage as a semiring?
Relationship to other Provenance Models

Lineage

“Real” Lineage

- Can we also model the list of sets of tuples lineage as a semiring?
- NO!
  - Assume existence of semiring $K_{RLin}$ that models lineage
  - Equivalent queries $q = R \cup S$ and $q' = S \cup R$
  - Assume tuple $t$ is in the result of $q/q'$ and was derived from $r_1$ and $s_1$
  - Lineage: $Lin(q, t) = \langle \{r_1\}, \{s_1\} \rangle \neq \langle \{s_1\}, \{r_1\} \rangle = Lin(q', t)$
  - Evaluation in $K_{RLin}$: $(q)(t) = r_1 + s_1 = s_1 + r_1 = (q')(t)$
  - $\Rightarrow$ no assumptions except that $K_{RLin}$ is semiring
  - $\Rightarrow K_{RLin}$ cannot exists
Perm Influence Contribution Semantics

**Semiring**

- Cannot exists for the same reason as Lineage
  - Assume existence of semiring \( K_{PI} \) that models PI-CS
  - Equivalent queries \( q = R \cup S \) and \( q' = S \cup R \)
  - Assume tuple \( t \) is in the result of \( q/q' \) and was derived from \( r_1 \) and \( s_1 \)
  - Lineage: \( PI(q, t) = \{ < r_1, s_1 > \} \neq \{ < s_1, r_1 > \} = PI(q', t) \)
  - Evaluation in \( K_{PI} \): \( (q)(t) = r_1 + s_1 = s_1 + r_1 = (q')(t) \)
  - \( \Rightarrow K_{PI} \) cannot exists
How-Provenance

Relationship to other Provenance Models

Perm Influence Contribution Semantics

Discussion

- Lineage and PI-CS consider the order of leaves in the algebra tree
- However, equivalent queries can have different orders
- If we abstract from the order, is the result expressible in the semiring model?
- **Rationale**: Define mapping $H$ from $\mathcal{PI}$ to $\mathbb{N}[I]$ that gets rid of the order
Relationship to other Provenance Models

Perm Influence Contribution Semantics

From \( \mathcal{PI} \) to \( \mathbb{N}[/] \)

- Witness-lists are basically \( \times \)
- The set of witness-lists is basically \( + \)

\[
H(\mathcal{PI}(q, t)) = \sum_{w \in \mathcal{PI}(q, t)} \prod_{i=1}^{n} w'[i]
\]

\[
w'[i] = \begin{cases} 
    w[i] & \text{if } w[i] \neq \bot \\
    1 & \text{otherwise}
\end{cases}
\]
Perm Influence Contribution Semantics

Example

\[ q = \pi_a(R) \cup (\pi_a(R \bowtie S)) \]
\[ \mathcal{PI}(q, t_1) = \{ < r_1, \bot >, < r_1, s_1 > \} \]
\[ H(\mathcal{PI}(q, t_1)) = \sum_{w \in \mathcal{PI}(q,t_1)} \prod_{i \in \{1, \ldots, n\}} w'[i] \]
\[ = r_1 \times 1 + r_1 \times s_1 = r_1 + r_1 \times s_1 \]
\[ = N[l](q, t_1) \]
Relationship to other Provenance Models

Relationships between Provenance Semirings

\[ (ab + b) = b \]
Outline

1. How-Provenance, Semirings, and Orchestra
   - Introduction
   - Semiring Semantics for Relational Algebra
   - How-Provenance or Provenance Polynomials
   - Relationship to other Provenance Models
   - ORCHESTRA
   - Recap
Overview

- Collaborative Data Sharing System
- Network of peers
- Each peer has independent schema and instance
- Peers update their instances without restrictions
- Schema mappings define relationships between schemata
  - Can be partial
- Periodically peers trigger exchange of updates based on mappings
Schema mappings

- **Schema mapping**: Logical constraints that define the relationship between two schemata.
- Different schema may store the same information in different structure.
- Schema mappings model these structures in the schema relate.
- With some extra mechanism can be use to translate data from one schema into the other.

**Example**

- **Schema** $S_1$: Person(Name, AddrId), Address(Id, City, Street)
- **Schema** $S_2$: LivesAt(Name, City)
ORCHESTRA

Update Exchange

- Each peer updates its instance as he pleases
- A log of update operations is kept
- Peers can trigger an update exchange

Update Exchange

- Determine updates since last exchange
- Translate updates from peers according to schema mappings
- Eagerly compute provenance during update exchange
Provenance in ORCHESTRA

- Use $\mathbb{N}[/]
- Add functions $m_1, \ldots, m_n$ to represent mappings
- E.g., $m_1(x + yz) + m_2(u)$ means that tuple was derived by
  - applying mapping $m_1$ to $x, y, z$
  - applying mapping $m_2$ to $u$
How Provenance Use in ORCHESTRA

Trust

- Instead of applying all updates: only apply “trusted” updates.
- Peers decide on a per mapping/peer basis whether they trust data.
  - Use Trust semiring: \((R_{\text{inf}}, \text{min}, +, \text{inf}, 0)\)
  - Evaluate provenance in the trust semiring using the trust value for peers and mappings.
Deletion Propagation

- Deletion in semiring model $\Rightarrow$ annotating with 0 element of semiring
- We have provenance for query result
- Assume set $D$ of tuples got deleted
- Set every occurrence of $D$ in the provenance of some tuple $t$ to 0
- Compute whether $t$ is still derivable
- Here even without index on provenance useful, because repeating whole update exchange is unfeasible
1. How-Provenance, Semirings, and Orchestra
   - Introduction
   - Semiring Semantics for Relational Algebra
   - How-Provenance or Provenance Polynomials
   - Relationship to other Provenance Models
   - ORCHESTRA
   - Recap
Recap

Semiring Semantics for the Relational Model

- Alternative semantics for relational algebra
- Given a semiring \((K, +, \times, 0, 1)\)
  - \(K\)-relations are functions from tuples of an arity \(U\) to semiring elements
  - Operators take functions (relations) as input and produce an output function (relation)
- Using different semirings we get standard semantics or extensions of the relational model
  - \(((\mathbb{B}, \lor, \land, false, true))\): Set semantics
  - \(((\mathbb{N}, +, \times, 0, 1))\): Bag semantics
  - \(((PosBool(X), \lor, \land, false, true))\): \(c\)-tables
How-Provenance Recap

How-Provenance (Provenance semiring)

- **Rationale**: Provenance for $t$ is expression that represent the semiring computation that lead to creation of tuple $t$.
- **Representation**: Polynomial over tuple variables ($=$ element of Provenance Semiring)
- **Syntactic Definition**:
  - For USPJ queries + extensions for A and D
- **The Fundamental Property**: Given an query result in $\mathbb{N}[I]$, we can compute the query result for any semiring $K$ from that
- **Relation to other Provenance Types**:
  - Semirings that model other provenance models
  - Why-Provenance: $(\mathcal{P}(\mathcal{P}(I)), \cup, \emptyset, \emptyset)$
  - IWhy-Provenance: $(\text{min}(\mathcal{P}(I)), \cup_{\text{min}}, \times_{\text{min}}, \emptyset, \emptyset)$
  - Lineage*: $(\mathcal{P}(I) \cup \{\bot\}, +, \times, \bot, \emptyset)$
 Recap

**ORCHESTRA**

- Peer-to-Peer update exchange system
- Schema mappings between peers
- Updates are exchanged between periodically based on mappings
- Provenance used for
  - Trust
  - Deletion propagation
## Provenance Model Comparison

<table>
<thead>
<tr>
<th>Property</th>
<th>Why</th>
<th>Lin</th>
<th>PI-CS</th>
<th>Where</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>Set of Set of Tuples</td>
<td>List of Set of Tuples</td>
<td>Set/Bag of List of Tuples</td>
<td>Sets of Attribute Value Positions</td>
<td>Values of provenance semiring</td>
</tr>
<tr>
<td>Granularity</td>
<td>Tuple</td>
<td>Tuple</td>
<td>Tuple</td>
<td>Attribute Value</td>
<td>Tuple</td>
</tr>
<tr>
<td>Language Support</td>
<td>USPJ</td>
<td>ASPJ-Set</td>
<td>ASPJ-Set + Nested sub-queries</td>
<td>U-SPJ</td>
<td>A<em>SPJ-UD</em></td>
</tr>
<tr>
<td>Semantics</td>
<td>Set</td>
<td>Set + Bag*</td>
<td>Bag</td>
<td>Set</td>
<td>Set + Bag</td>
</tr>
<tr>
<td>Variants</td>
<td>Wit, Why, IWhy</td>
<td>Set/Bag</td>
<td>Influence + Copy</td>
<td>SPJ + Insensitive + Insensitive Union</td>
<td>semirings</td>
</tr>
<tr>
<td>Design Principles</td>
<td>Sufficiency - No false positives</td>
<td>Sufficiency + No false negatives + no false positives</td>
<td>Sufficiency + No false negatives + No false positives</td>
<td>Copying</td>
<td>Equivalent to query evaluation</td>
</tr>
<tr>
<td>Systems</td>
<td>-</td>
<td>WHIPS</td>
<td>Perm</td>
<td>DBNotes</td>
<td>ORCHESTRA</td>
</tr>
<tr>
<td>Insensitivity</td>
<td>Yes - No - Yes</td>
<td>No</td>
<td>No</td>
<td>No - Yes - Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*ILLOINOIS INSTITUTE OF TECHNOLOGY*
Recap

Literature I

Yael Amsterdamer, Daniel Deutch, and Val Tannen.
On the limitations of provenance for queries with difference.
In Tapp '11: 3rd usenix workshop on the theory and practice of provenance, 2011.

Y. Amsterdamer, D. Deutch, T. Milo, and V. Tannen.
On provenance minimization.
In PODS '11, 2011.

Y. Amsterdamer, D. Deutch, and V. Tannen.
Provenance for Aggregate Queries.

T.J. Green, G. Karvounarakis, and Z.G.I.V. Tannen.
Provenance in ORCHESTRA.
, 2010.

Nicholas E. Taylor and Zachary G. Ives.
Reliable storage and querying for collaborative data sharing systems.
In ICDE '10, 2010.

G. Karvounarakis, Z.G. Ives, and V. Tannen.
Querying data provenance.
F. Geerts and A. Poggi.  
On database query languages for K-relations.  

Todd J. Green.  
Containment of Conjunctive Queries on Annotated Relations.  

Todd J. Green.  
Collaborative data sharing with mappings and provenance.  

Todd J. Green, Zachary G. Ives, and Val Tannen.  
Reconcilable differences.  
In ICDT ’09: Proceedings of the 16th International Conference on Database Theory, 212–224, Saint Petersburg, Russia, March 2009.

Zachary G. Ives, Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira.  
The ORCHESTRA Collaborative Data Sharing System.  

J. Nathan Foster, Todd J. Green, and Val Tannen.  
Annotated XML: Queries and Provenance.  
Recap

Literature III

Todd J. Green, Gregory Karvounarakis, and Val Tannen.
Provenance Semirings.

Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Olivier Biton, Zachary G. Ives, and Val Tannen.
ORCHESTRA: Facilitating Collaborative Data Sharing.

Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen.
Update Exchange with Mappings and Provenance.

Floris Geerts and Jan Van den Bussche.
Relational Completeness of Query Languages for Annotated Databases.

Zachary G. Ives, Nitin Khandelwal, Aneesh Kapur, and Murat Cakir.
ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data.