Combining dependent annotations for relational algebra

Egor V. Kostylev and Peter Buneman
(presented by Domagoj Vrgoč)

15th International Conference on Database Theory
March 27, 2012
Annotation – superimposing information on existing/base data
Major industry in **curated databases** – especially in bioinformatics

- **Uniprot/Swissprot** is a “database of annotation”
- **BioDAS** is called Distributed Annotation Server

Various attempts to formalise annotation and implement generic annotation systems (e.g., **DBNotes** by Chiticariu, et al).

**Natural question**: how should annotations propagate through a query on the base data?
This is obviously related to provenance.
Annotation – superimposing information on existing/base data

Major industry in **curated databases** – especially in bioinformatics

- **Uniprot/Swissprot** is a “database of annotation”
- **BioDAS** is called Distributed Annotation Server

Various attempts to formalise annotation and implement generic annotation systems (e.g., **DBNotes** by Chiticariu, et al).

**Natural question:** how should annotations propagate through a query on the base data?

This is obviously related to provenance.
Semirings for annotating databases
Examples of annotated databases and queries

(Usual) databases

\[ \text{Rel:} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
</tr>
</tbody>
</table>

\[ Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \Join \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \Join \pi_{BC}\text{Rel}) : \]

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
</tr>
</tbody>
</table>

Something similar?
Examples of annotated databases and queries
(Usual) databases

\[
\text{Rel:}
\begin{array}{ccc}
A & B & C \\
a & b & c \\
d & b & e \\
f & g & e \\
\end{array}
\]

\[
Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \Join \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \Join \pi_{BC}\text{Rel}):
\begin{array}{cc}
A & C \\
a & c \\
a & e \\
d & c \\
d & e \\
f & e \\
\end{array}
\]
Examples of annotated databases and queries

Temporal databases

\[ Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \bowtie \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \bowtie \pi_{BC}\text{Rel}) : \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>ValidTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>[1-4]</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>[2-5]</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>[3-8]</td>
</tr>
</tbody>
</table>

Something similar?
Examples of annotated databases and queries

Temporal databases

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>ValidTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>[1-4]</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>[1-4]</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>[2-5]</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>[3-8]</td>
</tr>
</tbody>
</table>

\(Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \Join \langle \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \Join \pi_{BC}\text{Rel}\rangle)\):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>ValidTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>([1-4] \cap [1-4] \cup ([1-4] \cap [1-4]) = [1-4])</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>([1-4] \cap [2-5] = [2-4])</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td>([1-4] \cap [2-5] = [2-4])</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>([2-5] \cap [2-5] \cup ([2-5] \cap [2-5] \cup ([2-5] \cap [3-8]) = [2-5])</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>([3-8] \cap [3-8] \cup ([3-8] \cap [3-8] \cup ([2-5] \cap [3-8]) = [3-8])</td>
</tr>
</tbody>
</table>

Something similar?
Examples of annotated databases and queries

Bag semantics

**Rel:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>5</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \Join \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \Join \pi_{BC}\text{Rel}) : \]

Something similar?
Examples of annotated databases and queries

Bag semantics

\[ Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \bowtie \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \bowtie \pi_{BC}\text{Rel}) : \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td>5</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>2 \cdot 2 + 2 \cdot 2 = 8</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>2 \cdot 5 = 10</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td>2 \cdot 5 = 10</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>5 \cdot 5 + 5 \cdot 5 + 5 \cdot 1 = 55</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>1 \cdot 1 + 1 \cdot 1 + 5 \cdot 1 = 7</td>
</tr>
</tbody>
</table>

Something similar?
Examples of annotated databases and queries

Bag semantics

**Rel:**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>e</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>e</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Q(\text{Rel}) = \pi_{AC}(\pi_{AB}\text{Rel} \bowtie \pi_{BC}\text{Rel} \cup \pi_{AC}\text{Rel} \bowtie \pi_{BC}\text{Rel}) : \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td></td>
<td>2 \cdot 2 + 2 \cdot 2 = 8</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td></td>
<td>2 \cdot 5 = 10</td>
</tr>
<tr>
<td>d</td>
<td>c</td>
<td></td>
<td>2 \cdot 5 = 10</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td></td>
<td>5 \cdot 5 + 5 \cdot 5 + 5 \cdot 1 = 55</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td></td>
<td>1 \cdot 1 + 1 \cdot 1 + 5 \cdot 1 = 7</td>
</tr>
</tbody>
</table>

Something similar?
**K-Relations [Green et al., 2007]**

- **K** – set with element 0 and operations $\oplus$ and $\otimes$
- **K-relation** – a mapping from a set of tuples (with the same schema) to $K$
- **Positive algebra on K-relations** has operations
  - **Union** $(R_1 \cup R_2)(t) = R_1(t) \oplus R_2(t)$
  - **Projection** $(\pi_V(R))(t) = \bigoplus_{t=t'} \bigoplus_{V} R(t')$, where $V$ is a set of attributes
  - **Selection** $(\sigma_P(R))(t) = R(t) \otimes P(t)$, where $P$ is a selection predicate
  - **Natural join** $(R_1 \bowtie R_2)(t) = R_1(t) \otimes R_2(t)$
  - **Renaming** $(\rho_\beta(R))(t) = R(t \circ \beta)$, where $\beta$ is a bijection
Semirings

Theorem (Green et al., 2007)

*Positive algebra on K-relations satisfies expected identities (associativity and commutativity of union and join, etc.) if and only if K is a commutative semiring.*

- **set semantics semiring**
  \[ \mathcal{B} = \langle \{ \text{false}, \text{true} \}, \lor, \land, \text{false}, \text{true} \rangle \] — a tuple is annotated with **true** if it is “in” the database, and **false** if it is not

- **temporal semiring**
  \[ \mathcal{T} = \langle 2^T, \cup, \cap, \emptyset, T \rangle \] — tuples are annotated with subsets of the set of time points \( T \)

- **bag semantics semiring**
  \[ \mathcal{N} = \langle \mathbb{N}_0, +, \times, 0, 1 \rangle \] — tuples are annotated with nonnegative numbers

- **probabilistic semiring**, **why-provenance semiring**, **linearge semiring**, **fuzzy semiring**, etc.
Combined dependent annotations
Multiple domains

• Domain of annotations for positive relational algebra (SPJU) is expected to be a semiring.
• What to do if we need to annotate a database with 2 domains $K_1$ and $K_2$?
  • Simple answer: annotations should be pairs from $K_1 \times K_2$, and operations should be applied pairwise to elements.
  • Does it always work?
Multiple domains

- Domain of annotations for positive relational algebra (SPJU) is expected to be a semiring.
- What to do if we need to annotate a database with 2 domains $K_1$ and $K_2$?
- Simple answer: annotations should be pairs from $K_1 \times K_2$, and operations should be applied pairwise to elements.
- Does it always work?
Multiple domains

- Domain of annotations for positive relational algebra (SPJU) is expected to be a semiring.
- What to do if we need to annotate a database with 2 domains $K_1$ and $K_2$?
- Simple answer: annotations should be pairs from $K_1 \times K_2$, and operations should be applied pairwise to elements.
- Does it always work?
Example

Exports:

<table>
<thead>
<tr>
<th>CName</th>
<th>Goods</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>Food</td>
<td>2004-2008</td>
<td>UK, Germany</td>
</tr>
<tr>
<td>Greece</td>
<td>Textile</td>
<td>2007-2010</td>
<td>Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

Time – sets of **years** with $\cup$ and $\cap$ as operations
Customers – sets of **countries** with $\cup$ and $\cap$ as operations

$$Q = \pi_{\text{CName}}(\text{Exports}) :$$

<table>
<thead>
<tr>
<th>CName</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>2004-2010</td>
<td>UK, Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

Is it the answer we expect?
Example

Exports:

<table>
<thead>
<tr>
<th>CName</th>
<th>Goods</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>Food</td>
<td>2004-2008</td>
<td>UK, Germany</td>
</tr>
<tr>
<td>Greece</td>
<td>Textile</td>
<td>2007-2010</td>
<td>Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

Time – sets of years with \( \cup \) and \( \cap \) as operations
Customers – sets of countries with \( \cup \) and \( \cap \) as operations

\[
Q = \pi_{\text{CName}}(\text{Exports}):
\]

<table>
<thead>
<tr>
<th>CName</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>2004-2010</td>
<td>UK, Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

Is it the answer we expect?
## Example

### Exports:

<table>
<thead>
<tr>
<th>CName</th>
<th>Goods</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>Food</td>
<td>2004-2008</td>
<td>UK, Germany</td>
</tr>
<tr>
<td>Greece</td>
<td>Textile</td>
<td>2007-2010</td>
<td>Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

- **Time** – sets of *years* with $\cup$ and $\cap$ as operations
- **Customers** – sets of *countries* with $\cup$ and $\cap$ as operations

$$Q = \pi_{CName}(\text{Exports}) :$$

<table>
<thead>
<tr>
<th>CName</th>
<th>Time</th>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greece</td>
<td>2004-2010</td>
<td>UK, Germany, Italy, Cyprus</td>
</tr>
</tbody>
</table>

Is it the answer we expect?
Graphical representation

([2004-2008], \{UK, Germany\})
([2007-2010], \{Germany, Italy, Cyprus\}):
Graphical representation

([2004-2010], {UK, Germany, Italy, Cyprus})
It is impossible to represent the desired set of dots by a single pair of elements from the combining domains.

Combined annotation – a set of pairs from $K_1 \times K_2$. 
It is impossible to represent the desired set of dots by a single pair of elements from the combining domains.

**Combined annotation** – a set of pairs from $K_1 \times K_2$. 
Example: Combined annotation

\[ \lambda_1 = \{(\{2004-2008\}, \{\text{UK, Germany}\}) \\
(\{2007-2010\}, \{\text{Germany, Italy, Cyprus}\})\}: \]
Example: Combined annotation

\[ \lambda_2 = \{ ([2004-2006], \{ \text{UK, Germany} \}) \,
\quad ([2007-2008], \{ \text{UK, Ger, Italy, Cyprus} \}) \} : \quad ([2009-2010], \{ \text{Germany, Italy, Cyprus} \}) \} : \]
Intuition: two combined annotations are equivalent if they “cover the same set of dots”

We should be careful, since some annotation semirings have neither negation nor idempotence!

However all of them (as far as we know) are naturally ordered and the underlying orders are lattices.
Equivalence on Combined Annotations: formally

- Define **immediate containment** $\preceq^*$ of combined annotations, using only lattice operations.

- Define **containment** $\preceq$ as the transitive closure of $\preceq^*$.

- Define **equivalence** $\sim$ as conjunction of $\preceq$ and $\succeq$. 
Equivalence on CombinedAnnotations: formally

- Define **immediate containment** $\preceq^*$ of combined annotations, using only lattice operations

- Define **containment** $\preceq$ as the transitive closure of $\preceq^*$
- Define **equivalence** $\sim$ as conjunction of $\preceq$ and $\succeq$
Semiring of (equivalence classes of) combined annotations

If we are given two semirings (annotation domains) $\mathcal{K}_1 = \left< K_1, \oplus_1, \otimes_1, 0_1, 1_1 \right>$ and $\mathcal{K}_2 = \left< K_2, \oplus_2, \otimes_2, 0_2, 1_2 \right>$ then combined domain is the semiring

$$\mathcal{C}^\sim [\mathcal{K}_1, \mathcal{K}_2] = \left< \mathcal{C}^\sim, \oplus, \otimes, 0, 1 \right>$$

where

- $\mathcal{C}^\sim$ is the set of equivalence classes of combined annotations
- $\oplus$ constructed on the base of $\oplus_1$ and $\oplus_2$ and lattice operations (such that $\lambda \oplus \mu$ always greater than $\lambda$ and $\mu$)
- $\otimes$ constructed on the base of $\otimes_1$ and $\otimes_2$ and lattice operations
- $0 = (0_1, 0_2)^\sim$
- $1 = (1_1, 1_2)^\sim$
Normal form of combined annotations
Normal form of combined annotations

We would like a computable normal form for equivalence classes to effectively store and compare combined annotations, again on the base only lattice operations.

We need to be careful again!

One more restriction, satisfied by all (known) annotation domains: the underlying lattices should be distributive.
Normal form of combined annotations

We would like a computable normal form for equivalence classes to effectively store and compare combined annotations, again on the base only lattice operations.

We need to be careful again!

One more restriction, satisfied by all (known) annotation domains: the underlying lattices should be distributive.
Normal form of combined annotations

We would like a computable \textbf{normal form} for equivalence classes to effectively store and compare combined annotations, again on the base only lattice operations.

We need to be careful again!

One more restriction, satisfied by all (known) annotation domains: the underlying lattices should be distributive.
Example of normal form

\[ \lambda = \{([2004-2008], \{\text{UK, Germany}\}) \}
\]

\[ (\[2007-2010\], \{\text{Germany, Italy, Cyprus}\})\} : \]
Example of normal form

\[\text{Norm}(\lambda) = \{([2004-2008], \{\text{UK, Germany}\}) \]
\([2007-2010], \{\text{Germany, Italy, Cyprus}\})\]
\([2007-2008], \{\text{UK, Ger, Italy, Cyprus}\})\]
\([2004-2010], \{\text{Germany}\})\}::

\[
\begin{array}{cccccccc}
\text{UK} & & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\text{Germ} & & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\text{Ita} & & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
\text{Cyp} & & & & & & & \\
& & & & & & & \\
& & & & & & & \\
& & & & & & & \\
\end{array}
\]

\[
\begin{array}{cccccccc}
'04 & '05 & '06 & '07 & '08 & '09 & '10 \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array}
\]
Normal form always exists, finite and unique.

We designed an algorithm to compute the normal form.

Not always practical: the size may be exponentially large.
Computing normal form

Normal form always exists, finite and unique.

We designed an algorithm to compute the normal form.

Not always practical: the size may be exponentially large.
Computing normal form

**Normal form** always exists, finite and unique.

We designed an **algorithm** to compute the normal form.

Not always practical: the size may be exponentially large.
In the paper we generalize this construction to $N$ annotation domains, such that pairs of them are

- either independent (behave as cartesian products for sums)
- or dependent (behave as in the example).
Future work

We have concentrated on the principles of combined annotations and their normalization. It would be useful

- to have complexity bounds for deciding containment and equivalence for certain classes of combined domains;
- to extend the approach to over operations of relational algebra, e.g., negation.
Thank you!