Provenance for Natural Language Queries

Daniel Deutch, Nave Frost, Amit Gilad – Tel Aviv University
VLDB 2017
Presented By: Dor Ma’ayan
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
Consider the Following Database

<table>
<thead>
<tr>
<th>Rel. org</th>
<th>Rel. author</th>
</tr>
</thead>
<tbody>
<tr>
<td>oid</td>
<td>oname</td>
</tr>
<tr>
<td>1</td>
<td>UPENN</td>
</tr>
<tr>
<td>2</td>
<td>TAU</td>
</tr>
<tr>
<td>5</td>
<td>Slava N.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rel. pub</th>
<th>Rel. writes</th>
</tr>
</thead>
<tbody>
<tr>
<td>wid</td>
<td>cid</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rel. conf</th>
<th>Rel. domainConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>cid</td>
<td>cname</td>
</tr>
<tr>
<td>10</td>
<td>SIGMOD</td>
</tr>
<tr>
<td>11</td>
<td>VLDB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rel. domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
</tr>
<tr>
<td>18</td>
</tr>
</tbody>
</table>
Motivation

NL Interfaces to database systems has been the focus of multiple lines of research recently.

**NL Query**
Return the organization of authors who published papers in database conferences after 2005.

**Formal Query**

```prolog
query(oname) :- org(oid, oname), conf(cid, cname), pub(wid, cid, ptitle, pyear), author(aid, aname, oid), domainConf(cid, did), domain(did, dname), writes(aid, wid), dname = 'Databases', pyear > 2005
```
Motivation

The result of the query is simple

Formal Query

query(oname) :- org(oid, oname), conf(cid, cname),
pub(wid, cid, ptitle, pyear), author(aid, aname, oid),
domainConf(cid, did), domain(did, dname),
writes(aid, wid), dname = 'Databases', pyear > 2005

Result

Tel Aviv University (TAU)
Motivation

The result of the query is simple

**Formal Query**

```prolog
query(oname) :- org(oid, oname), conf(cid, cname),
pub(wid, cid, ptitle, pyear), author(aid, aname, oid),
domainConf(cid, did), domain(did, dname),
writes(aid, wid), dname = 'Databases', pyear > 2005
```

**Result**

Tel Aviv University (TAU)
We Want: Explanation & Justification!

- Providing NL explanation to query answers
- Elaborate upon answers with additional important information
  - Why does each answer qualify to the query criteria?

What We Want - Explanations

TAU is the organization of 43 authors who published 170 papers in 31 conferences in 2006 - 2015
### Solution Overview - Provenance

#### What We Have - Provenance

```plaintext
(cname, SIGMOD)·(pyear, 14')+
(cname, VLDB)·(pyear, 06')+
(cname, VLDB)·(pyear, 07')+
(cname, SIGMOD)·(pyear, 14')+
(cname, SIGMOD)·(pyear, 14')+
...
```
Outline

- Introduction & Motivation
- **Dependency Trees**
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
A dependency tree $T = (V,E,L)$ is a node-labeled tree where labels consist of two components, as follows:

1. **Part of Speech (POS):** the syntactic role of the word
2. **Relationship (REL):** the grammatical relationship between the word and its parent in the dependency tree.
NL Query

Return the organization of authors who published papers in database conferences after 2005
Abstract Dependency Trees

Verb Mod

Non-Verb mod
Abstract Dependency Trees

Non-Verb mod
From Dependency Tree to CQ by NaLIR

Formal Query

query(oname) :- org(oid, oname), conf(cid, cname), pub(wid, cid, ptitle, pyear), author(aid, aname, oid), domainConf(cid, did), domain(did, dname), writes(aid, wid), dname = 'Databases', pyear > 2005
And Now Formally...

Given a dependency tree $T = (V, E, L)$ and a CQ $Q$, a dependency-to-query-mapping:

$$\tau : V \rightarrow \text{Vars}(Q)$$

is a partial function mapping a subset of the dependency tree nodes to the variables of $Q$. 

Formal Query

query(oracle) :- org(oid, oname), conf(cid, cnamr), pub(wid, cid, ptitle, pyear), author(aid, aname, oid), domainConf(cid, did), domain(did, dname), writes(aid, wid), dname = 'Databases', pyear > 2005
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
  - From Provenance to NL: Single Assignment
  - From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
Provenance - Reminder

What We Have - Provenance

(o, TAU)·(a, Tova M.)·(p, OASSIS...)·(c, SIGMOD)·(y, 14′)+
(o, TAU)·(a, Tova M.)·(p, Querying...)·(c, VLDB)·(y, 06′)+
(o, TAU)·(a, Tova M.)·(p, Monitoring...)·(c, VLDB)·(y, 07′)+
(o, TAU)·(a, Slava N.)·(p, OASSIS...)·(c, SIGMOD)·(y, 14′)+
(o, TAU)·(a, Tova M.)·(p, A sample...)·(c, SIGMOD)·(y, 14′)+
...
Assignment for CQs

An assignment $\alpha$ for a query $Q \in CQ$ with respect to a database instance $D$ is a mapping of the relational atoms of $Q$ to tuples in $D$ that respects relation names and induces a mapping over variables/constants.

Assignments allow for defining the semantics of CQs:

A tuple $t$ is said to appear in the query output if there exists an assignment $\alpha$ s.t. $t = \alpha(head(Q))$. 
Assignment: Example

Formal Query

query(oname) :- org(oid, oname), conf(cid, cname),
pub(wid, cid, ptitle, pyear), author(aid, aname, oid),
domainConf(cid, did), domain(did, dname),
writes(aid, wid), dname = 'Databases', pyear > 2005
From Assignments to Provenance

The idea: assignments capture the reasons for a tuple to appear in the query result, with each assignment serving as an alternative such reason.

Let \( A(Q,D) \) be the set of assignments for a CQ \( Q \) and a database instance \( D \). We define the value-level provenance of \( Q \) w.r.t. \( D \) as:

\[
\sum_{\alpha \in A(Q,D)} \prod_{\{x_i, a_i | \alpha(x_i) = a_i\}} (x_i, a_i)
\]
Provenance

$$\sum_{\alpha \in A(Q,D)} \prod\{x_i, a_i | \alpha(x_i) = a_i\}(x_i, a_i)$$

**Formal Query**

```sql
query(oname) :- org(oid, oname), conf(cid, cname),
pub(wid, cid, ptitle, pyear), author(aid,aname, oid),
domainConf(cid, did), domain(did, dname),
writes(aid, wid), dname = 'Databases', pyear > 2005
```

**What We Have - Provenance**

- (oname,TAU)::(aname,Tova M.)::(ptitle,OASSIS...)::(cname,SIGMOD)::(pyear,14')+
- (oname,TAU)::(aname,Tova M.)::(ptitle,Querying...)::(cname,VLDB)::(pyear,06')+
- (oname,TAU)::(aname,Tova M.)::(ptitle,Monitoring...)::(cname,VLDB)::(pyear,07')+
- (oname,TAU)::(aname,Slava N.)::(ptitle,OASSIS...)::(cname,SIGMOD)::(pyear,14')+
- (oname,TAU)::(aname,Tova M.)::(ptitle,A sample...)::(cname,SIGMOD)::(pyear,14')+

**Relation: org**

<table>
<thead>
<tr>
<th>aid</th>
<th>oname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UPENN</td>
</tr>
<tr>
<td>2</td>
<td>TAU</td>
</tr>
</tbody>
</table>

**Relation: author**

<table>
<thead>
<tr>
<th>aid</th>
<th>aname</th>
<th>oid</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Susan D.</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Tova M.</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Slava N.</td>
<td>2</td>
</tr>
</tbody>
</table>

**Relation: pub**

<table>
<thead>
<tr>
<th>wid</th>
<th>cid</th>
<th>ptitle</th>
<th>pyear</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
<td>&quot;OASSIS...&quot;</td>
<td>2014</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>&quot;A sample...&quot;</td>
<td>2014</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>&quot;Monitoring...&quot;</td>
<td>2007</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>&quot;Querying...&quot;</td>
<td>2006</td>
</tr>
</tbody>
</table>

**Relation: writes**

<table>
<thead>
<tr>
<th>aid</th>
<th>wid</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

**Relation: conf**

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>SIGMOD</td>
</tr>
<tr>
<td>11</td>
<td>VLDB</td>
</tr>
</tbody>
</table>

**Relation: domainConf**

<table>
<thead>
<tr>
<th>cid</th>
<th>did</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
</tr>
</tbody>
</table>

**Relation: domain**

<table>
<thead>
<tr>
<th>did</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Databases</td>
</tr>
</tbody>
</table>
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- **From Provenance to NL: Single Assignment**
- From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
We now describing the transformation of provenance to NL

First, we will handle the case where the query has a single assignment with respect to the input database

Later on, we will use that simple solution as a building block for the more complex case

What We Have - Provenance

\[
\begin{align*}
& (\text{oname}, \text{TAU}) \cdot (\text{aname}, \text{Tova M.}) \cdot (\text{ptitle}, \text{OASSIS...}) \cdot (\text{cname}, \text{SIGMOD}) \cdot (\text{pyear}, 14') + \\
& (\text{oname}, \text{TAU}) \cdot (\text{aname}, \text{Tova M.}) \cdot (\text{ptitle}, \text{Querying...}) \cdot (\text{cname}, \text{VLDB}) \cdot (\text{pyear}, 06') + \\
& (\text{oname}, \text{TAU}) \cdot (\text{aname}, \text{Tova M.}) \cdot (\text{ptitle}, \text{Monitoring...}) \cdot (\text{cname}, \text{VLDB}) \cdot (\text{pyear}, 07') + \\
& (\text{oname}, \text{TAU}) \cdot (\text{aname}, \text{Slava N.}) \cdot (\text{ptitle}, \text{OASSIS...}) \cdot (\text{cname}, \text{SIGMOD}) \cdot (\text{pyear}, 14') + \\
& (\text{oname}, \text{TAU}) \cdot (\text{aname}, \text{Tova M.}) \cdot (\text{ptitle}, \text{A sample...}) \cdot (\text{cname}, \text{SIGMOD}) \cdot (\text{pyear}, 14') + \\
& \ldots
\end{align*}
\]
Overview

Follow the structure of the NL query dependency tree and generate an answer tree with the same structure by replacing/modifying the words in the question with the values from the result and provenance that were mapped using the dependency-to-query-mapping and the assignment.
But That’s Not Enough!
**Compute Answer Tree - Algorithm**

- **Input:** A dependency tree, an empty answer tree, a mapping, and a *single assignment*.

- **Output:** An answer tree.

- **Idea:** Follow the structure of the NL query dependency tree and generate an answer tree with the same structure by replacing/modifying the words in the question with the values from the result and provenance that were.

```
class ComputeAnswerTree:
    def __init__(self, Q_dependency, A_tree, mapping, single_assignment):
        self.Q_dependency = Q_dependency
        self.A_tree = A_tree
        self.mapping = mapping
        self.single_assignment = single_assignment

    def get_answer_tree(self):
        result = self.compute_answer_tree(self.Q_dependency, self.A_tree, self.mapping, self.single_assignment)
        return result

def compute_answer_tree(Q_dependency, A_tree, mapping, single_assignment):
    # Implementation of the algorithm
    pass
```

---

**Algorithm 1: ComputeAnswerTree**

```
input : A dependency tree $T_Q$, an answer tree $T_A$
(empty in the first call), a dependency-to-query-mapping $\tau$, an assignment
$\alpha$, a node object $\in T_Q$

output: Answer tree with explanations $T_A$

1. child := null;
2. if object is a leaf then
   3. value = $\alpha(\tau(object))$;
   4. Replace(object, value, $T_A$);
3. else if $L(object).REL$ is mod then
   5. value = $\alpha(\tau(child_{T_Q}(object))$;
   6. Replace(tree(object), value, $T_A$);
   7. AddParent($T_A$, value);
8. else if object has a child $v$ s.t. $L(v).REL \in MOD$ and
    $L(v).POS \notin VERB$ then
   9. Adjust($T_Q$, $T_A$, $\tau$, $\alpha$, object, false);
   10. child := $v$;
11. else if object has a child $v$ s.t. $L(v).REL \in MOD$ and
    $L(v).POS \in VERB$ then
12. Adjust($T_Q$, $T_A$, $\tau$, $\alpha$, object, true);
13. child := $v$;
14. if child $\neq$ null then
15.   foreach $u \in children_{T_Q}(child)$ do
16.     ComputeAnswerTree($T_Q$, $T_A$, $\tau$, $\alpha$, $u$);
17. return $T_A$;
```
Compute Answer Tree

Starting with the query dependency tree

(oname, TAU)·(aname, Tova M.)·(ptitle, OASSIS...)·(cname, SIGMOD)·(pyear, 14')+
Compute Answer Tree Cont.

(oname, TAU)·(aname, Tova M.)·(ptitle, OASSIS...)·(cname, SIGMOD)·(pyear, 14′)+
Compute Answer Tree Cont.

\[(\text{oname, TAU}) \leftarrow \text{organization} \quad \text{(POS=NN, REL=doobj)}\]

\[\leftarrow \text{the} \quad \text{(POS=IN, REL=prep)}\]

\[\text{of} \quad \text{authors} \quad \text{(POS=NNS, REL=pobj)}\]

\[\text{published} \quad \text{(POS=VBD, REL=rcmd)}\]

\[\text{who} \quad \text{papers} \quad \text{(POS=IN, REL=prep)}\]

\[\text{after} \quad \text{conferences} \quad \text{(POS=NNS, REL=pobj)}\]

\[\text{in} \quad \text{2005} \quad \text{(POS=CD, REL=pobj)}\]

\[\text{database} \quad \text{(POS=NN, REL=nn)}\]

\[\text{(oname,TAU)·(aname,Tova M.)·(ptitle,OASSIS...)·(cname,SIGMOD)·(pyear,14')}\]
Compute Answer Tree Cont.

(ontitle, TAU) \cdot (aname, Tova M.) \cdot (ptitle, OASSIS...) \cdot (cname, SIGMOD) \cdot (pyear, 14') +
Compute Answer Tree Cont.

(oname, TAU) • (aname, Tova M.) • (ptitle, ‘OASSIS...’) • (pyear, 2014) • (cname, SIGMOD)
Answer

TAU is the organization of Tova M. who published 'OASSIS...' in SIGMOD in 2014
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
The General Case

- We have considered the case where the provenance consists of a single assignment.
- Let’s generalize our solution to multiple assignments:
  - Naïve solution: Generate a sentence for each individual assignment and concatenating the resulting sentences.
  - This solution is bad: this would result in a long and unreadable answers.
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- **Factorization**
  - Factorization to Answer Tree
  - Implementation & Experiments
Provenance Factorization - Idea

Use algebraic factorization of the provenance to take-out common values that appear in multiple assignments.
Factorization - Definition

Let $P$ be a provenance expression. We say that an expression $f$ is a factorization of $P$ if $f$ may be obtained from $P$ through (repeated) use of some of the following axioms:

- **Distributivity of summation over multiplication**
- **Associativity of both summation and multiplication.**
- **Commutativity of both summation and multiplication.**
### Factorization Examples

#### Provenance

<table>
<thead>
<tr>
<th>Source</th>
<th>Publication</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>[TAU] [Tova M.]</td>
<td>[OASSIS...][SIGMOD]</td>
<td>2014</td>
</tr>
<tr>
<td>[TAU]</td>
<td>[Querying...][VLDB]</td>
<td>2006</td>
</tr>
<tr>
<td>[TAU] [Tova M.]</td>
<td>[Monitoring...][VLDB]</td>
<td>2007</td>
</tr>
<tr>
<td>[TAU] [Slava N.]</td>
<td>[OASSIS...][SIGMOD]</td>
<td>2014</td>
</tr>
<tr>
<td>[TAU] [Tova M.]</td>
<td>[A sample...][SIGMOD]</td>
<td>2014</td>
</tr>
</tbody>
</table>

#### Two Different Factorizations

<table>
<thead>
<tr>
<th>Source</th>
<th>Publication</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>[TAU]</td>
<td>[SIGMOD] [2014]</td>
<td></td>
</tr>
<tr>
<td>([OASSIS...])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>([Tova M.] + [Slava N.])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ [Tova M.] [A Sample...])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ [VLDB] [Tova M.]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>([2006] [Querying...])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ [2007] [Monitoring...])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ [SIGMOD] [2014]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>([OASSIS...]) + [A Sample...])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ [Slava N.] [OASSIS...][SIGMOD] [2014]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Shortest Factorization

\[
[\text{TAU}] \cdot \\
(\text{SIGMOD}) \cdot [2014] \cdot \\
(\text{OASSIS...}) \cdot \\
((\text{Tova M.}) + [\text{Slava N.}]) \cdot \\
+ [\text{Tova M.}] \cdot [\text{A Sample...}] \cdot \\
+ \text{VLDB} \cdot [\text{Tova M.}] \cdot \\
([2006] \cdot [\text{Querying...}] \cdot \\
+ [2007] \cdot [\text{Monitoring...}])
\]

### As a Sentence

TAU is the organization of authors who published in SIGMOD 2014 'OASSIS...' which was published by Tova M. and Slava N. and Tova M. published 'A sample...' and Tova M. published in VLDB 'Querying...' in 2014 and 'Monitoring...' in 2007.
We factorize as a step towards generating an NL answer.

We want the (partial) order of the nesting of the value annotations in the factorization is consistent with the (partial) order of corresponding words in the NL query.

Partial order of nodes in dependency tree:

Given an dependency tree $T$, we define $\leq_T$ as the descendant partial order of nodes in $T$: for each two nodes, $x, y \in V(T)$, we say that $x \leq_T y$ if $x$ is a descendant of $y$ in $T$. 
Partial Order of Provenance

- let $T$ be a query dependency tree
- let $\text{prov}_T$ be a provenance expression
- let $f$ be a factorization of $\text{prov}_T$
- let $\tau$ be a dependency-to-query-mapping
- let $\{\alpha_1, ..., \alpha_n\}$ be the set of assignments to the query.

For each two nodes $x, y$ in $T$ we say that $x \leq_f y$ if
$$\forall i \in [n]: \text{level}_f(\alpha_i(\tau(x))) \leq \text{level}_f(\alpha_i(\tau(y))).$$
T-compatible

We say that $f$ is T-compatible if each pair of nodes $x \neq y \in V(T)$ that satisfy $x \leq_T y$ also satisfy that $x \leq_f y$.

Let $T$ be an NL query dependency tree and let $prov_T$ be a provenance expression for the answer. We say that a factorization $f$ of $prov_T$ is optimal if $f$ is T-compatible and there is no T-compatible factorization $f'$ of $prov_T$ such that $|f'| < |f|$ (where $|f|$ is the length of $f$).
Example

\[ \text{Shortest Factorization} \]

\[
[\text{TAU}] \cdot \\
([\text{SIGMOD}] \cdot [2014]) \cdot \\
([\text{OASSIS}]) \cdot \\
([\text{Tova M.}] + [\text{Slava N.}]) \\
+ [\text{Tova M.}] \cdot [\text{A Sample...}] \\
+ [\text{VLDB}] \cdot [\text{Tova M.}] \cdot \\
([2006] \cdot [\text{Querying...}] \\
+ [2007] \cdot [\text{Monitoring...}])
\]

\[ \text{conferences} \leq_{T} \text{authors} \]

But

\[ \text{conferences} \not\leq_{f_{bad}} \text{authors} \]
Obtaining a minimal $T$-compatible factorization is coNP-hard
Factorization Algorithm

- Factorize greedily: traverse the dependency tree level-by-level
- For every level with mapped words, factorize their corresponding values in the provenance
- Prioritize which values to take-out in each level by frequency

Complexity

$O(n^2 \cdot \log n)$: recursively traverse the dependency tree and sort the variables at each layer by their frequency in $O(n \cdot \log n)$

```
Algorithm 2: GreedyFactorization

input: $T_Q$ - the query tree, $\leq_{T_Q}$ - the query partial order, prov - the provenance, $\tau, \alpha$ - dependency-to-query-mapping and assignment from nodes in $T_Q$ to provenance variables, Processed - subset of nodes from $V(T_Q)$ which were already processed (initially, $\emptyset$)
output: $f$ - $T_Q$-compatible factorization of $\text{pro}v_{T_Q}$

1. $f \leftarrow \text{prov}$;
2. $\text{Frontier} \leftarrow \{x \in V(T_Q) | V(y \in V(T_Q) \setminus \text{Processed}) \text{ s.t. } x \nsubseteq T_Q y\}$;
3. $\text{vars} \leftarrow \text{sortByFrequentVars}(\{\alpha(\tau(x)) | x \in \text{Frontier}\}, f)$;
4. foreach $\text{var} \in \text{vars}$ do
5.   Take out $\text{var}$ from sub-expressions in $f$ not including variables from $\{x | \exists y \in \text{Processed} : x = \alpha(\tau(y))\}$;
6. $\text{Processed} \leftarrow \text{Processed} \cup \text{Frontier}$;
7. if $|\text{Processed}| = |V(T_Q)|$ then
8.   return $f$;
9. else
10.  return $\text{GreedyFactorization}(T_Q, f, \tau, \alpha, \text{Processed})$;
```
Factorization Example
Factorization Example

\[ \text{organization} \quad \text{of} \quad \text{authors} \quad \text{published} \quad \text{after} \quad 2005 \quad \text{conferences} \quad \text{database} \]

\[ \text{[TAU]} \cdot ([\text{Tova M.}] \cdot [OASSIS...] \cdot [SIGMOD] \cdot [2014] + [\text{Tova M.}] \cdot [Querying...] \cdot [VLDB] \cdot [2006] + [\text{Tova M.}] \cdot [Monitoring...] \cdot [VLDB] \cdot [2007] + [\text{Slava N.}] \cdot [OASSIS...] \cdot [SIGMOD] \cdot [2014] + [\text{Tova M.}] \cdot [A sample...] \cdot [SIGMOD] \cdot [2014]) \]
Factorization Example

[TAU].

([Tova M.].

([OASSIS...]. [SIGMOD]. [2014] +

[Querying...]. [VLDB]. [2006] +

[Monitoring...]. [VLDB]. [2007] +

[A sample...]. [SIGMOD]. [2014]) +

[Slava N.]. [OASSIS...]. [SIGMOD]. [2014])
Factorization Example

[TAU] ·
([Tova M.] ·
([VLDB] ·
  ([2006] · [Querying...] + [2007] · [Monitoring...])))
  + [SIGMOD] · [2014] ·
    ([OASSIS...] + [A Sample...])
    + [Slava N.] · [OASSIS...] · [SIGMOD] · [2014])
TAU is the organization of Tova M. who published in VLDB 'Querying...' in 2006 and 'Monitoring...' in 2007 and in SIGMOD in 2014 'OASSIS...' and 'A sample...' and Slava N. who published 'OASSIS...' in SIGMOD in 2014.
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- Factorization
  - Factorization to Answer Tree
- Implementation & Experiments
Factorization to Answer Tree

Algorithm 3: ComputeFactAnswerTree

input : $\alpha$ - an assignment to the NL query, $T_A$ - answer
dependency tree based on $\alpha$, root - the root of
the circuit induced by the factorized provenance

output: $T_F$ - tree of the factorized answer

1 $T_F \leftarrow \text{copy}(T_A);$
2 foreach $p \in \text{children}_f(\text{root})$ do
3     if $p$ is a leaf then
4         $\text{val} \leftarrow \alpha(\text{var}(p));$
5         $\text{node} \leftarrow \text{Lookup}(\text{var}(p), \alpha, T_A);$
6         $\text{ReplaceVal}(\text{val}, \text{node}, T_F);$
7         $\text{Rearrange}(\text{node}, T_A, T_F);$
8     else
9         $T_F^{\text{rec}} \leftarrow \text{ComputeFactAnswerTree}(\alpha, T_A, p);$
10        $\text{RecNodes} = V(T_A) \setminus V(T_A);$
11        $\text{parent}_{F}^{\text{rec}} \leftarrow \text{LCA}(\text{recNodes});$
12        $\text{parent}_F \leftarrow \text{Corresponding node to} \text{parent}_{F}^{\text{rec}} \text{in} T_F;$
13        $\text{Attach} \text{recNodes to} T_F \text{ under the parent}_F;$
14 return $T_F;$
Factorization to Answer Tree Example

TAU is the organization of
Tova M. who published
in VLDB
‘Querying...’ in 2006 and
‘Monitoring...’ in 2007
and in SIGMOD in 2014
‘OASSIS...’ and ‘A sample...’
and Slava N. who published
‘OASSIS...’ in SIGMOD in 2014.

UPENN is the organization of Susan D. who published
‘OASSIS...’ in SIGMOD in 2014.
From Factorization to Summarization

- When there are many assignments and / or the assignments involve multiple distinct values, even an optimal factorization representation may be too long.

- To this end, we employ summarization, as follows:
  - First, we note that a key to summarization is understanding which parts of the provenance may be grouped together.
  - Nodes are of the same type if both were mapped to the same query variable.
  - Now, let n be a node in the circuit form of a given factorization f. A summarization of the sub-circuit of n is obtained in two steps:
    - First, we group the descendants of n according to their type.
    - Then, we summarize each group separately.
Summarization

Two Levels of Summarization

TAU

A

{ ([Tova M.] · ([VLDB] · ([2006] · [Querying...]
+ [2007] · [Monitoring...])))
+ [SIGMOD] · [2014] · ([OASSIS...] + [A Sample...])
+ [Slava N.] · [OASSIS...] · [SIGMOD] · [2014])

B

Shorter Summarized Answer Based on A

TAU is the organization of 2 authors who published
4 papers in 2 conferences in 2006 - 2014

More Detailed Summarized Answer Based on B

TAU is the organization of Tova M. who published
4 papers in 2 conferences in 2006 - 2014 and Slava N.
who published ’OASSIS...’ in SIGMOD in 2014.
Outline

- Introduction & Motivation
- Dependency Trees
- Provenance
- From Provenance to NL: Single Assignment
- From Provenance to NL: The General Case
- Factorization
- Factorization to Answer Tree
- Implementation & Experiments
System Architecture
## Sample Use Cases

<table>
<thead>
<tr>
<th>Query</th>
<th>Single Assignment</th>
<th>Multiple Assignments - Summarized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the homepage of SIGMOD</td>
<td><a href="http://www.sigmod2011.org/">http://www.sigmod2011.org/</a> is the homepage of SIGMOD</td>
<td>Tova M. published 10 papers in SIGMOD in 2006-2014</td>
</tr>
<tr>
<td>Return the authors who published papers in SIGMOD before 2015 and after 2005</td>
<td>Tova M. published “Auto-completion...” in SIGMOD in 2012</td>
<td>Tova M. from TAU published 11 papers in VLDB</td>
</tr>
<tr>
<td>Return the authors from TAU who published papers in VLDB</td>
<td>Tova M. from TAU published “XML Repository...” in VLDB</td>
<td>Tova M. published 96 papers in 18 conferences</td>
</tr>
<tr>
<td>Return the authors who published papers in database conferences</td>
<td>Tova M. “published Auto-completion...” in SIGMOD</td>
<td>Tova M. published 170 papers in 31 conferences in 2006 - 2015</td>
</tr>
<tr>
<td>Return the organization of authors who published papers in database conferences after 2005</td>
<td>TAU is the organization of Tova M. who published ‘OASSIS...’ in SIGMOD in 2014</td>
<td>TAU is the organization of 43 authors who published 170 papers in 31 conferences in 2006 - 2015</td>
</tr>
</tbody>
</table>
# User Study

<table>
<thead>
<tr>
<th>Category</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relevant 4</td>
<td>4</td>
<td>10</td>
<td>84</td>
<td>4.82</td>
</tr>
<tr>
<td>Understandable</td>
<td>7</td>
<td>25</td>
<td>66</td>
<td>4.60</td>
</tr>
<tr>
<td><strong>Multiple</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relevant 0</td>
<td>0</td>
<td>7</td>
<td>43</td>
<td>4.86</td>
</tr>
<tr>
<td>Understandable</td>
<td>4</td>
<td>13</td>
<td>33</td>
<td>4.58</td>
</tr>
<tr>
<td>Detailed 3</td>
<td>3</td>
<td>7</td>
<td>40</td>
<td>4.74</td>
</tr>
<tr>
<td><strong>Summarized</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relevant 2</td>
<td>2</td>
<td>2</td>
<td>46</td>
<td>4.88</td>
</tr>
<tr>
<td>Understandable</td>
<td>3</td>
<td>3</td>
<td>44</td>
<td>4.82</td>
</tr>
<tr>
<td>Detailed 2</td>
<td>2</td>
<td>5</td>
<td>43</td>
<td>4.82</td>
</tr>
</tbody>
</table>
## Scalability

<table>
<thead>
<tr>
<th>Query</th>
<th>Query Eval. Time</th>
<th>Fact. Time</th>
<th>Sentence Gen. Time</th>
<th>NLProv Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.9</td>
<td>0.038</td>
<td>0.096</td>
<td>0.134</td>
</tr>
<tr>
<td>5</td>
<td>0.6</td>
<td>0.03</td>
<td>0.14</td>
<td>0.17</td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>0.62</td>
<td>2.08</td>
<td>2.7</td>
</tr>
<tr>
<td>7</td>
<td>20.5</td>
<td>1.1</td>
<td>3.1</td>
<td>4.2</td>
</tr>
<tr>
<td>8</td>
<td>2.4</td>
<td>0.001</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>9</td>
<td>0.01</td>
<td>0.011</td>
<td>0.001</td>
<td>0.012</td>
</tr>
<tr>
<td>10</td>
<td>21.3</td>
<td>0.53</td>
<td>2.23</td>
<td>2.76</td>
</tr>
<tr>
<td>11</td>
<td>53.7</td>
<td>3.18</td>
<td>6.46</td>
<td>9.64</td>
</tr>
<tr>
<td>12</td>
<td>18.8</td>
<td>3.22</td>
<td>1.73</td>
<td>4.95</td>
</tr>
</tbody>
</table>
Sample Scalability Results

Computation time as a function of the number of assignments. Overhead of only 16% w.r.t evaluation time.
Limitations

- Part of the solution was designed to fit NaLir, and will need to be replaced if a different NL query is used.

- The solution is limited to Conjunctive Queries, handling other structures is challenging and will require modifications to the definition of provenance.
Thank You
Questions?