On Factorisation of Provenance Polynomials

Dan Olteanu and Jakub Závodný

Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF OXFORD

Provenance Polynomials

Unifying framework (Green et al.) that captures the semantics of
- incomplete information and uncertain databases,
- query evaluation under set/bag semantics,
- annotation propagation for why- and how-provenance.

In provenance polynomials, we denote provenance of
- input tuples by **variables**,
- a join of tuples by a **product** of their provenance,
- a union of tuples by a **sum** of their provenance.

Example Database

Order

	id	item
o_1	01	Printer
o_2	02	Plotter
o_3	03	Ink
o_4	04	Printer
o_5	05	Ink

Store

	location	item
s_1	Depot1	Printer
s_2	Depot1	Plotter
s_3	Depot2	Printer
s_4	StoreA	Ink

Emp

	operator	location
e_1	Joe	Depot1
e_2	Bob	Depot1
e_3	Dan	Depot2
e_4	Dan	StoreA

Example Query

Order $\bowtie_{\text{item}}$ Store $\bowtie_{\text{location}}$ Emp

	id	item	location	operator
$o_1 s_1 e_1$	01	Printer	Depot1	Joe
$o_1 s_1 e_2$	01	Printer	Depot1	Bob
$o_1 s_3 e_3$	01	Printer	Depot2	Dan
$o_2 s_2 e_1$	02	Plotter	Depot1	Joe
...			...	

Provenance Polynomial of the Query Result

$$\begin{aligned}\Phi_1 = & o_1 s_1 e_1 + o_1 s_1 e_2 + o_1 s_3 e_3 + o_2 s_2 e_1 + o_2 s_2 e_2 + \\ & o_3 s_4 e_4 + o_4 s_1 e_1 + o_4 s_1 e_2 + o_4 s_3 e_3 + o_5 s_4 e_4.\end{aligned}$$

Special cases:
- Boolean semiring $(\mathbb{B}, \vee, \wedge)$
  - Each variable encodes the presence of its input tuple.
  - Used in incomplete information and probabilistic databases.
- Semiring over natural numbers $(\mathbb{N}, +, \bullet)$
  - Each variable encodes tuple multiplicity.
  - Used in bag semantics of positive queries.
- If the variables encode the tuples themselves, the provenance polynomial encodes the whole query result.

Factorisation of Provenance Polynomials

Algebraic factorisation of Φ_1:

$$\Phi_2 = (o_1 + o_4)(s_1(e_1 + e_2) + s_3 e_3) + o_2 s_2 (e_1 + e_2) + (o_3 + o_5) s_4 e_4.$$

expresses explicitly how groups of input tuples combine and thus shows the nested structure of the query result and its provenance.

- Factorisations can be **more informative** and **exponentially more succint** than flat representations.
- The monomials can be extracted from the factorisation **with polynomial delay**.

Challenge: Queries with Factorised Polynomials of Bounded Size

Classification of queries based on
- the minimal size of the factorised polynomials of query results for any input database
- result polynomials with factorisations of **bounded readability** for any input database
  - Polynomial Φ is **read-k** if each variable occurs at most k times in Φ.
  - Polynomial Φ has **readability** k if k is the smallest number such that there is a read-k polynomial equivalent to Φ.

Examples: The readability of
- the query [Store $\bowtie_{\text{location}}$ Emp] is one for any database.
  - In our example, the factorised polynomial is $(s_1 + s_2)(e_1 + e_2) + s_3 e_3 + s_4 e_4$.
  - For each location, we get a product of sums of distinct variables.
- the query [Order $\bowtie_{\text{item}}$ Store $\bowtie_{\text{location}}$ Emp] is dependent on the input database size.

Challenge: Efficient Computation of Factorised Polynomials

- Compute factorisations of low/minimal readability for any polynomial.
  - Minimality may be with respect to a restricted class of factorisations.
- For a query and a database, compute the factorised polynomial of the query result
  - **without** first computing the flat polynomial of the query result.

Challenge: Querying Factorised Relations and Polynomials

- Assume that variables in polynomials carry the input tuples.
- Evaluate queries **directly** on factorised polynomials.

Example: Equivalent factorisations of the result of [Order $\bowtie_{\text{item}}$ Store $\bowtie_{\text{location}}$ Emp]:

$$\Phi_9 = (o_1 + o_2)(s_1(e_1 + e_2) + s_2(e_3 + e_4)) + (o_3 + o_4)(s_3(e_1 + e_2) + s_4(e_3 + e_4)),$$
$$\Phi_{10} = ((o_1 + o_2)s_1 + (o_3 + o_4)s_3)(e_1 + e_2) + ((o_1 + o_2)s_2 + (o_3 + o_4)s_4)(e_3 + e_4).$$

- Here, variables $o_i(e_j)$ are annotated with tuples from Order (Emp)
- $\Phi_9(\Phi_{10})$ is suitable for joining on Order (Emp) without unfolding

Challenge: Approximation by Factorised Polynomials

Given a polynomial Φ, find **lower** and **upper bounds** Φ_L, Φ_U with lower readability.
- Definition of lower and upper bounds depends on the semiring.
  - In the Boolean semiring: $\Phi_L \models \Phi \models \Phi_U$
  - In the semiring over natural numbers: $\Phi_L \leq \Phi \leq \Phi_U$
  - For all semirings: Drop (add) monomials for lower (upper) bounds

  Lower bound for Φ_1: $\Phi_L = (o_1 + o_4)(s_1(e_1 + e_2) + s_3 e_3) + (o_3 + o_5) s_4 e_4$

  Upper bound for Φ_1: $\Phi_U = (o_1 + o_2 + o_4)((s_1 + s_2)(e_1 + e_2) + s_3 e_3) + (o_3 + o_5) s_4 e_4.$
- Search for closest bounds in a given class $\mathcal{C}$ of well factorisable polynomials.
  - $\mathcal{C}$ could be the class of polynomials with readability one.

Query approximation:
- Approximate a query Q by lower and upper bound queries Q_L and Q_U.
  - For any database, the polynomials Φ_L and Φ_U of Q_L and Q_U are lower and upper bounds for the polynomial Φ of Q and have lower readability.

Results: Queries with Factorisations of Bounded Size

- We introduce **factorisation trees** which
  - are statically derived from a query Q,
  - are independent of the input database,
  - define a factorisation of the polynomial of $Q(\mathbf{D})$, for any database $\mathbf{D}$.

Characterisation of Conjunctive Queries

- For any query Q, there is a rational number $f(Q)$ such that for any database $\mathbf{D}$, $Q(\mathbf{D})$ has a factorised polynomial
  - with readability $O(|\mathbf{D}|^{f(Q)})$,
  - with size at most $|\mathbf{D}|^{f(Q)+1}$.

  Moreover, $f(Q)$ is the smallest such number when restricted to factorisations defined by factorisation trees.
- A query satisfies $f(Q) = 0$ iff it is **hierarchical**. Then the polynomial of any $Q(\mathbf{D})$ has a factorisation
  - with **bounded readability**,
  - with size linear in the sizes of input database and query.

  For hierarchical queries w/o self-joins it is also known that
  - in probabilistic databases, their exact probability can be computed in polynomial time,
  - in the finite cursor machine model, they can be evaluated in just one pass over the database.

Results: Efficient Computation of Factorisations

- For any Q and $\mathbf{D}$, we compute a factorisation of readability $O(|\mathbf{D}|^{f(Q)})$ and size at most $|\mathbf{D}|^{f(Q)+1}$ in time $O(|\mathbf{D}|^{f(Q)+1})$
- .. without computing the flat polynomial!

Results: Approximation by Factorised Polynomials

Approximation by polynomials of readability one, over the Boolean semiring.
- Equivalent syntactic and model-theoretic characterisations of lower and upper bounds.
- Algorithms to enumerate bounds with polynomial delay.

Selected Publications

- **On Factorisation of Provenance Polynomials**
  D. Olteanu and J. Závodný. In *TaPP*, 2011.
- **Factorised Representations of Query Results.**
  D. Olteanu and J. Závodný. Tech. rep., Oxford, April 2011. Also arXiv report 1104.0867.
- **On the Optimal Approximation of Queries Using Tractable Propositional Languages.**
  R. Fink and D. Olteanu. In *ICDT*, 2011.