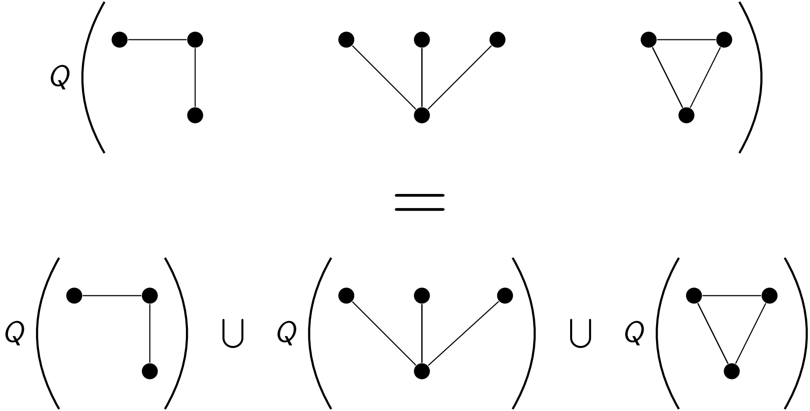
Additive First-Order Queries

Gerald Berger [1] Martin Otto [2] Andreas Pieris [3] Dimitri Surinx [4]
Jan Van den Bussche [4]

[1]TU Wien [2]TU Darmstadt [3]U. Edinburgh [4]U. Hasselt

Problem statement

Characterize the FO queries that can be answered by only looking at domain-disjoint subinstances



Instances & Queries

A *fact* has the form $R(a_1, \ldots, a_k)$ where R/k is a relation name and $a_1, \ldots, a_k$ come from an infinite universe of data elements

A *database schema* S is a finite set of relation names

An *instance* I is a nonempty set of facts over S

The *active domain* $adom(I)$ is set of all data elements in I

A k-ary query over S maps each instance I over S to a k-ary subset on $adom(I)$

Additivity

A query Q is additive if $Q(I \cup J) = Q(I) \cup Q(J)$ for any instances I, J such that $adom(I) \cap adom(J) = \emptyset$

We denote the class of additive queries with ADD

Interesting class of queries:

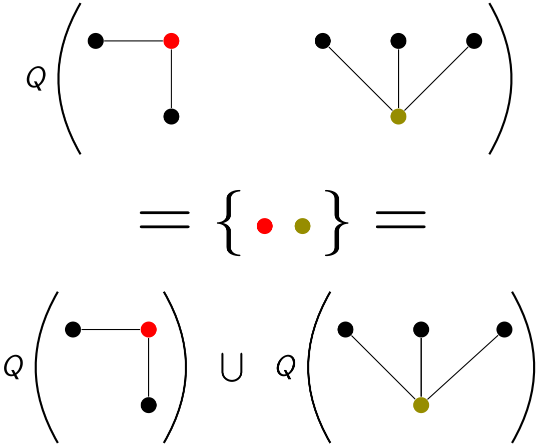
- Allows coordination-free distributed query evaluation
- Useful in the analysis of the expressiveness of query languages

Examples:

- Computing the union and difference of relations are additive
- Computing the cartesian product is **NOT** additive

An additive query

$Q =$ "Select the nodes with degree at least 2"



Expressible in FO: $\{(x) \mid \exists y \exists z((R(x,y) \wedge R(x,z)) \wedge y \neq z)\}$

An additive query qanguage: connected FO (CFO)

CFO is a syntactical fragment of FO:

- Conjunction $\varphi \wedge \psi$ is only allowed when $\mathrm{free}(\varphi) \cap \mathrm{free}(\psi) \neq \emptyset$
- Disjunction $\varphi \vee \psi$ is only allowed when $\mathrm{free}(\varphi) = \mathrm{free}(\psi)$
- Negation is only allowed in the form $\varphi \wedge \neg\psi$ where $\mathrm{free}(\psi) \neq \emptyset$, and $\mathrm{free}(\psi) \subseteq \mathrm{free}(\varphi)$
- For all quantification abbreviations are not allowed

Example: $\varphi(x) = \exists y \exists z((R(x, y) \wedge R(x, z)) \wedge \neg(y = z))$ is in CFO

Connected FO

Theorem

$\mathrm{FO} \cap \mathrm{ADD} = \mathrm{CFO}$

$\boxed{\supseteq}$: CFO is additive

$\boxed{\subseteq}$: Different proofs for:

- Formula with at least one free variable
- Sentences

The theorem holds over all instances, and over finite instances

$\mathrm{FO} \cap \mathrm{ADD} = \mathrm{CFO}$ for formulas

Lemma

Every additive FO formula φ is $2^{\mathsf{qr}(\varphi)}$-local

A formula $\varphi(\bar{x})$ is called ℓ-*local* if for every instance I and every tuple $\bar{a}$:

$$\bar{a} \in \varphi(I) \Leftrightarrow \bar{a} \in \varphi(N^I(\bar{a}, \ell))$$

where $N^{\mathcal{A}}(\bar{a}, \ell)$ is the restriction of I to the ball $B(\bar{a}, \ell)$

$\mathrm{FO} \cap \mathrm{ADD} = \mathrm{CFO}$ for formulas

Lemma

Every additive FO formula φ is $2^{\mathrm{qr}(\varphi)}$-local

Lemma

Let $\varphi(x_1, \ldots, x_n)$ be an additive formula. For every $(a_1, \ldots, a_n) \in \varphi(I)$ we have $d^I(a_i, a_j) \leq (n-1)2^{\mathrm{qr}(\varphi)}$.

Both Lemmas are proven using an Ehrenfeucht-Fraïssé game argument inspired by Otto's work on bisimulation invariance in modal logic

$\mathrm{FO} \cap \mathrm{ADD} = \mathrm{CFO}$ for formulas

Lemma

Every additive FO formula φ is $2^{\mathsf{qr}(\varphi)}$-local

- $\rightarrow$ we can relativize every quantifier in φ to $B(\mathrm{free}(\varphi), 2^{\mathsf{qr}(\varphi)})$

Lemma

Let $\varphi(x_1, \ldots, x_n)$ be an additive formula. For every $(a_1, \ldots, a_n) \in \varphi(I)$ we have $d^I(a_i, a_j) \leq (n-1)2^{\mathsf{qr}(\varphi)}$.

- $\rightarrow$ $\varphi \equiv \varphi \wedge \delta$ where $\delta = \bigwedge_{1 \leq i,j \leq n} d(x_i, x_j) \leq (n-1)2^{\mathsf{qr}(\varphi)}$
- $\rightarrow$ Can easily be converted to a CFO formula by pushing in δ

$\mathrm{FO} \cap \mathrm{ADD} = \mathrm{CFO}$ for sentences

Proposition

Every additive FO sentence φ can be rewritten as a finite disjunction of simple local sentences

A sentence of the form $\exists x \psi(x)$ where ψ is local, is called simple local

Proof idea: Rewrite φ as $\exists x \varphi^*(x)$ where φ^* is $(x = x) \wedge \varphi$

- φ^* is invariant under disjoint copies ($\bar{a} \in \varphi^*(I)$ iff $\bar{a} \in \varphi^*(I \cup \mathrm{cop}(I))$)
  - $\Rightarrow$ φ is equivalent to a boolean combination (in DNF) of simple local sentences (Otto's work on bisimulation invariance in modal logic)
- Eliminate negations
- Eliminate conjunctions

Example: eliminating negations

Consider an additive query $\sigma_1 \wedge \neg\sigma_2$ where σ_1 and σ_2 are additive

There exists domain-disjoint instances I and J such that:

- $I \models \sigma_1 \wedge \neg\sigma_2$
- $J \models \sigma_2$

$$
\begin{tikzcd}
I \models \sigma_1 \wedge \neg\sigma_2 \arrow[r, "additivity"] & I \cup J \models \sigma_1 \wedge \neg\sigma_2 \arrow[d, dashed, red, "contradiction"] \\
J \models \sigma_2 \arrow[r, "additivity"] & I \cup J \models \sigma_2
\end{tikzcd}
$$

Example: eliminating conjunctions

Consider an additive query $\sigma_1 \wedge \sigma_2$ where σ_1 and σ_2 are additive

There exists domain-disjoint instances I and J

- $I \models \sigma_1$ and $I \not\models \sigma_2$
- $J \models \sigma_2$ and $J \not\models \sigma_1$

$$
\begin{tikzcd}
I \models \sigma_1 \arrow[r, "additivity"] & I \cup J \models \sigma_1 \arrow[d, dash] & & I \models \sigma_1 \wedge \sigma_2 \\
& \text{and} \arrow[r] & I \cup J \models \sigma_1 \wedge \sigma_2 \arrow[r, "additivity"] & \text{or} \arrow[u, dashed, "contradiction"'] \arrow[d, dashed, "contradiction"] \\
J \models \sigma_2 \arrow[r, "additivity"] & I \cup J \models \sigma_2 \arrow[u, dash] & & J \models \sigma_1 \wedge \sigma_2
\end{tikzcd}
$$

Consequences

Connected relational algebra (CRA) is a sublanguage of RA where:

- Cartesian product is not allowed
- Equijoins are allowed

Corollary

$$\mathrm{RA} \cap \mathrm{ADD} = \mathrm{CRA}$$

Guarded fragment (GF)

Define CGF as CFO formulas that are guarded

Theorem

$$\mathrm{GF} \cap \mathrm{ADD} = \mathrm{CGF}$$

Proof idea: If $\varphi \in \mathrm{GF} \cap \mathrm{ADD}$

1. φ can be written as a boolean combination (in DNF) of CGF formulas
2. negated formulas with one free variable and positive formulas within each clause are connected
3. In each clause, the free variables of the negative part are included in those of the positive part
4. Clauses cannot contain negative sentences
5. All clauses have the same free variables

Example: positive formulas are connected

Consider $\sigma_1 \wedge \sigma_2$ where σ_1, σ_2 are additive and $\mathrm{free}(\sigma_1) \cap \mathrm{free}(\sigma_2) = \emptyset$

There exists domain-disjoint instances I and J:

- Exists valuation α on I: $I, \alpha \models \sigma_1$
- Exists valuation β on J: $J, \beta \models \sigma_2$
- $I \cup J, \xi \not\models \sigma_1 \wedge \sigma_2$ where ξ equals to α on $\mathrm{free}(\sigma_1)$ and to β on $\mathrm{free}(\sigma_2)$

$$\begin{tikzcd}
I, \alpha \models \sigma_1 \arrow[r, "additivity"] & I \cup J, \alpha \models \sigma_1 \arrow[r] & I \cup J, \xi \models \sigma_1 \arrow[d, dash] \\
 & I \cup J, \xi \models \sigma_1 \wedge \sigma_2 & \text{and} \arrow[l, dashed, red, "contradiction"'] \\
J, \beta \models \sigma_2 \arrow[r, "additivity"] & I \cup J, \beta \models \sigma_2 \arrow[r] & I \cup J, \xi \models \sigma_2 \arrow[u, dash]
\end{tikzcd}$$

Example: free variables of negative part $\rightarrow$ positive part

Consider $\sigma_1 \wedge \neg\sigma_2$ where σ_1, σ_2 are additive and that there exists x in $\mathrm{free}(\sigma_2)$ that is not in $\mathrm{free}(\sigma_1)$

There exists domain-disjoint instances I and J:

- exists valuation α on I: $I, \alpha \models \sigma_1 \wedge \neg\sigma_2$
- J, β is a domain-disjoint copy of I, α

Define valuation ξ that is equal to β on x and equal to α otherwise

$$\begin{tikzcd}
I, \alpha \models \sigma_1 \arrow[r] & I, \xi \models \sigma_1 \arrow[r, "\textit{additivity}"] & I \cup J, \xi \models \sigma_1 \arrow[d, no head] \\
\xi \text{ mixed for } \mathrm{free}(\sigma_2) \arrow[r, "\textit{additivity}"] & I \cup J, \xi \models \neg\sigma_2 \arrow[r, no head] & \text{and} \arrow[d] \\
\xi \text{ mixed for } \mathrm{free}(\sigma_1 \wedge \neg\sigma_2) & & I \cup J, \xi \models \sigma_1 \wedge \neg\sigma_2 \arrow[ll, dashed, red, "\textit{contradiction}"]
\end{tikzcd}$$

Complexity of additivity in GF

Theorem

Additivity of guarded formulas is 2ExpTime*-complete*

Hardness: reduction from satisfiability for guarded formulas, which is 2ExpTime-hard:

$$\varphi \text{ is unsatisfiable} \Leftrightarrow \varphi \wedge \exists x S(x) \wedge \exists y T(y) \text{ is additive}$$

Membership: polynomial reduction to satisfiability for guarded formulas

- Known that satisfiability is decidable in $2^{O(n) \cdot 2^{a \log a}}$ time for formulas of size n and maximum arity a
- Our reduction preserves arity

Reduction to satisfiability

Add two unary relation names U_1 and U_2 to the schema S

- Use U_1 and U_2 to mimic domain-disjoint instances

Consider instances I where the U_1 -and U_2-facts partition I into domain-disjoint instances (I_1, I_2)

- Definable by a GF formula φ_{cons}

For a GF formula φ define GF formulas ψ_i for $i = 1, 2$:

- Relativize the free variables and quantifiers in φ to U_i

$\rightarrow$ $(\psi_1 \vee \psi_2)(I) = \varphi(I_1) \cup \varphi(I_2)$

Lemma

$$\varphi \textit{ is additive} \Leftrightarrow \varphi_{cons} \wedge \neg(\varphi \leftrightarrow (\psi_1 \vee \psi_2)) \textit{ is unsatisfiable}$$

Positive Existential (PE)

Fragment of FO where only $\exists$, $\wedge$ and $\vee$ are allowed

Theorem (earlier work)

$$\mathrm{PE} \cap \mathrm{ADD} = \mathrm{CUCQ}$$

Deciding additivity for UCQs is NP-complete (earlier work)

Translation from PE to UCQ takes exponential time

- Gives an exponential upper bound for additivity checking

We have shown a more precise upper bound

Theorem

Additivity of PE formulas is Π_2^P-complete. The lower bound holds even over unary and binary relations.

Future work

- Investigate the complexity of the shortest connected formula equivalent to an additive formula
- Our hardness proof for additivity in GF is not specific to GF. Interesting to apply the reduction to other decidable fragments
- Our version for additivity is very strict in the context of distributed computation
  - Interesting to consider other ways to partition instances