The Complexity of Weighted Counting for Acyclic Conjunctive Queries

Arnaud Durand\textsuperscript{1}  Stefan Mengel\textsuperscript{2}

\textsuperscript{1}Univ Paris Diderot

\textsuperscript{2}University of Paderborn
Acyclic Conjunctive Queries
Acyclic Conjunctive Queries

Hardness results for counting
Outline

- Acyclic Conjunctive Queries
- Hardness results for counting
- Quantified star size
Outline

Acyclic Conjunctive Queries

Hardness results for counting

Quantified star size
Conjunctive queries I

- some logical relations over a domain $D$
Conjunctive queries I

- some logical relations over a domain $D$
- first order \{$\land, \exists$\}-formula $\phi$
Conjunctive queries I

- some logical relations over a domain $D$
- first order $\{\land, \exists\}$-formula $\phi$
- examples: $G = (V, E)$ graph, set $D := V$, relation $E$ is edge relation
some logical relations over a domain $D$

first order $\{\land, \exists\}$-formula $\phi$

examples: $G = (V, E)$ graph, set $D := V$, relation $E$ is edge relation

triangle in $G$:

$$v_1 \neq v_2 \land v_1 \neq v_3 \land v_2 \neq v_3 \land E(v_1, v_2) \land E(v_1, v_3) \land E(v_2, v_3)$$
Conjunctive queries I

- some logical relations over a domain $D$
- first order $\{\land, \exists\}$-formula $\phi$
- examples: $G = (V, E)$ graph, set $D := V$, relation $E$ is edge relation
  - triangle in $G$:
    \[ v_1 \neq v_2 \land v_1 \neq v_3 \land v_2 \neq v_3 \land E(v_1, v_2) \land E(v_1, v_3) \land E(v_2, v_3) \]
  - $k$-clique:
    \[ \bigwedge_{i,j \in [k], i \neq j} (E(v_i, v_j) \land v_i \neq v_j) \]
Conjunctive queries I

- some logical relations over a domain $D$
- first order $\{\land, \exists\}$-formula $\phi$
- examples: $G = (V, E)$ graph, set $D := V$, relation $E$ is edge relation
  - triangle in $G$:
    $$v_1 \neq v_2 \land v_1 \neq v_3 \land v_2 \neq v_3 \land E(v_1, v_2) \land E(v_1, v_3) \land E(v_2, v_3)$$
  - $k$-clique:
    $$\bigwedge_{i, j \in [k], i \neq j} (E(v_i, v_j) \land v_i \neq v_j)$$
  - vertices $v$ and $w$ connected by $k$-walk:
    $$\exists u_1 \exists u_2 \ldots \exists u_{k-1} E(v, u_1) \land \bigwedge_{i \in [k-2]} E(u_i, u_{i+1}) \land E(u_{k-1}, w)$$
different perspectives on conjunctive queries
different perspectives on conjunctive queries

- studied in AI literature as constraint satisfaction problems
 Conjunctive queries II

different perspectives on conjunctive queries
- studied in AI literature as constraint satisfaction problems
- important class of database queries, equivalent to select-project-join queries
different perspectives on conjunctive queries

- studied in AI literature as constraint satisfaction problems
- important class of database queries, equivalent to select-project-join queries
- equivalent to homomorphism problem in finite model theory
different perspectives on conjunctive queries

- studied in AI literature as constraint satisfaction problems
- important class of database queries, equivalent to select-project-join queries
- equivalent to homomorphism problem in finite model theory
- can be used to encode many different combinatorial problems
Different algorithmic questions

- Are there any answers to the query? Boolean Conjunctive query problem, BCQ.
Different algorithmic questions

- Are there any answers to the query? Boolean Conjunctive query problem, BCQ.
- Enumerate the answers to the query.
Different algorithmic questions

- Are there any answers to the query? Boolean Conjunctive query problem, BCQ.
- Enumerate the answers to the query.
  - may be exponentially many
Different algorithmic questions

- Are there any answers to the query? Boolean Conjunctive query problem, BCQ.
- Enumerate the answers to the query.
  - may be exponentially many
  - output polynomial time, polynomial delay, ...
Different algorithmic questions

- Are there any answers to the query? Boolean Conjunctive query problem, BCQ.
- Enumerate the answers to the query.
  - may be exponentially many
  - output polynomial time, polynomial delay, ...
- How many answers to the query are there?
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

\[ Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g) \]
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

\[ Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g) \]
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

\[
Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g)
\]

\[
\bullet e
\]

\[
\bullet f \quad \bullet d
\]

\[
\bullet g
\]

\quad

\[
\bullet a \quad \bullet b \quad \bullet c
\]
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

\[ Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g) \]
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

$$Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g)$$
Conjunctive Queries and their hypergraphs

- general case is NP-hard (reformulation of SAT)
- structural tractable subclasses, depending on the structure of the formula
- associate hypergraph to formula

\[ Q(a, b, c) \land R(a, e, f) \land R(c, d, e) \land P(a, c, e, g) \]
Acyclic hypergraphs

- organize the edges of a hypergraph into a “join tree”
Acyclic hypergraphs

- organize the edges of a hypergraph into a “join tree”
organize the edges of a hypergraph into a "join tree"
Acyclic hypergraphs

- organize the edges of a hypergraph into a “join tree”

- with a connectivity condition: edges that contain vertex $v$ form subtree
Acyclic hypergraphs

- organize the edges of a hypergraph into a “join tree”

- with a connectivity condition: edges that contain vertex \( v \) form subtree

- hypergraph is acyclic if it has a join tree
Conjunctive queries is acyclic, if its hypergraph is acyclic
Conjunctive queries is acyclic, if its hypergraph is acyclic
deciding acyclicity and constructing join trees is tractable
Acyclic Conjunctive Queries (ACQ)

- Conjunctive queries is acyclic, if its hypergraph is acyclic
- deciding acyclicity and constructing join trees is tractable
- Boolean Acyclic Conjunctive query problem is tractable [Yannakakis 81]
Conjunctive queries is acyclic, if its hypergraph is acyclic
deciding acyclicity and constructing join trees is tractable
Boolean Acyclic Conjunctive query problem is tractable [Yannakakis 81]
also: enumerating answers to ACQ can be done with polynomial delay [Bagan, Durand, Grandjean 07]
Acyclic Conjunctive Queries (ACQ)

- Conjunctive queries is acyclic, if its hypergraph is acyclic
- deciding acyclicity and constructing join trees is tractable
- Boolean Acyclic Conjunctive query problem is tractable [Yannakakis 81]
- also: enumerating answers to ACQ can be done with polynomial delay [Bagan, Durand, Grandjean 07]
- many generalizations to “nearly acyclic” queries (treewidth, cliquewidth, hypertreewidth, ...)
Outline

Acyclic Conjunctive Queries

Hardness results for counting

Quantified star size
count the answers to conjunctive queries
count the answers to conjunctive queries
for general conjunctive queries clearly intractable (#P-hard)
- count the answers to conjunctive queries
- for general conjunctive queries clearly intractable (\#P-hard)
- quantifier free \#ACQ: counting tractable [Pichler, Skritek 2011]
- count the answers to conjunctive queries
- for general conjunctive queries clearly intractable (\#P-hard)
- quantifier free \#ACQ: counting tractable [Pichler, Skritek 2011]
- algorithm similar to well known decision algorithms
count the answers to conjunctive queries
for general conjunctive queries clearly intractable (\#P-hard)
quantifier free \#ACQ: counting tractable [Pichler, Skritek 2011]
algorithm similar to well known decision algorithms
for decision $\exists$-quantifiers do not change the complexity
count the answers to conjunctive queries
for general conjunctive queries clearly intractable (\(\#P\)-hard)
quantifier free \(#\text{ACQ}\): counting tractable [Pichler, Skritek 2011]
algorithm similar to well known decision algorithms
for decision \(\exists\)-quantifiers do not change the complexity
counting problem gets hard with \(\exists\)-quantifiers
Quantified \#ACQ is hard [Pichler, Skritek 2011]

Theorem

\#ACQ with \(\exists\)-quantifiers is \#P-complete.
Quantified $\#ACQ$ is hard [Pichler, Skritek 2011]

**Theorem**

$\#ACQ$ with $\exists$-quantifiers is $\#P$-complete.

- $\#ACQ$ clearly in $\#P$ (guess assignment to the free variables, plug them in, solve remaining BACQ instance on the quantified variables with standard algorithm)
Theorem

\#ACQ with \(\exists\)-quantifiers is \#P-complete.

- \#ACQ clearly in \#P (guess assignment to the free variables, plug them in, solve remaining BACQ instance on the quantified variables with standard algorithm)
- \#ACQ is \#P-hard already with a single \(\exists\)-quantifier and simple structure
Theorem

\#ACQ with \(\exists\)-quantifiers is \#P-complete.

- \#ACQ clearly in \#P (guess assignment to the free variables, plug them in, solve remaining BACQ instance on the quantified variables with standard algorithm)
- \#ACQ is \#P-hard already with a single \(\exists\)-quantifier and simple structure
Quantified #ACQ is hard [Pichler, Skritek 2011]

Theorem

#ACQ with $\exists$-quantifiers is #P-complete.

- #ACQ clearly in #P (guess assignment to the free variables, plug them in, solve remaining BACQ instance on the quantified variables with standard algorithm)
- #ACQ is #P-hard already with a single $\exists$-quantifier and simple structure

Can we find tractable subclasses?
Outline

Acyclic Conjunctive Queries

Hardness results for counting

Quantified star size
Quantified star size

Ideas:

- prevent "big stars"
- disconnected quantified variables can be treated independently
- quantified component: maximal set of edges that is connected
- quantified star size: size of biggest independent set of free variables in any quantified component (here 3)
- huge for hard instance of [Pichler, Skritek 2011]
Quantified star size

Ideas:

▶ prevent “big stars”
Quantified star size

Ideas:

- prevent “big stars”
- disconnected quantified variables can be treated independently
Quantified star size

Ideas:

- prevent “big stars”
- disconnected quantified variables can be treated independently
Quantified star size

Ideas:

- prevent “big stars”
- disconnected quantified variables can be treated independently

quantified, free

- quantified component: maximal set of edges that is connected by quantified variables

[pictorial representation]
Quantified star size

Ideas:

- prevent “big stars”
- disconnected quantified variables can be treated independently

**quantified, free**

- quantified component: maximal set of edges that is connected by quantified variables
- quantified star size: size of biggest independent set of free variables in any quantified component (here 3)
Quantified star size

Ideas:

▶ prevent “big stars”
▶ disconnected quantified variables can be treated independently

quantified, free

▶ quantified component: maximal set of edges that is connected by quantified variables
▶ quantified star size: size of biggest independent set of free variables in any quantified component (here 3)
▶ huge for hard instance of [Pichler, Skritek 2011]
Theorem

For $\#\text{ACQ}$ of quantified star size $k$ one can count the answers in time $n^{O(k)}$. 

How good is this result?

▶ Can we compute quantified star size? YES

▶ Can we count much faster, e.g. fixed parameter tractable? NO

▶ Are there better parameters, i.e. larger tractable subclasses of $\#\text{ACQ}$? NO
Theorem

For \( \#ACQ \) of quantified star size \( k \) one can count the answers in time \( n^{O(k)} \).

How good is this result?
Theorem

For \( \#ACQ \) of quantified star size \( k \) one can count the answers in time \( n^{O(k)} \).

How good is this result?

- Can we compute quantified star size?
Theorem

For $\#ACQ$ of quantified star size $k$ one can count the answers in time $n^{O(k)}$.

How good is this result?

- Can we compute quantified star size? YES
Theorem

For \#ACQ of quantified star size $k$ one can count the answers in time $n^{O(k)}$.

How good is this result?

- Can we compute quantified star size? YES
- Can we count much faster, e.g. fixed parameter tractable?
Theorem

For \( \#ACQ \) of quantified star size \( k \) one can count the answers in time \( n^{O(k)} \).

How good is this result?

- Can we compute quantified star size?  YES
- Can we count much faster, e.g. fixed parameter tractable?  NO
Theorem

For \#ACQ of quantified star size $k$ one can count the answers in time $n^{O(k)}$.

How good is this result?

- Can we compute quantified star size? YES
- Can we count much faster, e.g. fixed parameter tractable? NO
- Are there better parameters, i.e. larger tractable subclasses of \#ACQ?
Theorem

For \#ACQ of quantified star size k one can count the answers in time \( n^{O(k)} \).

How good is this result?

- Can we compute quantified star size? YES
- Can we count much faster, e.g. fixed parameter tractable? NO
- Are there better parameters, i.e. larger tractable subclasses of \#ACQ? NO
Computing star size

- suffices to compute maximum independent set of acyclic hypergraphs

[Guo, Niedermeier 2006]
Computing star size

- suffices to compute maximum independent set of acyclic hypergraphs
- show version of König’s Lemma: for acyclic hypergraphs minimum edge covers and maximum independent sets have same size
Computing star size

- suffices to compute maximum independent set of acyclic hypergraphs
- show version of König’s Lemma: for acyclic hypergraphs minimum edge covers and maximum independent sets have same size
- minimum edge covers easy to compute [Guo, Niedermeier 2006]
Computing star size

- suffices to compute maximum independent set of acyclic hypergraphs
- show version of König’s Lemma: for acyclic hypergraphs minimum edge covers and maximum independent sets have same size
- minimum edge covers easy to compute [Guo, Niedermeier 2006]
- modification of the algorithm computes independent set, too
Fixed parameter tractability?

easy problems:
Fixed parameter tractability?

- **Easy problems:**
  - Counting problem $F : \{0, 1\}^* \times \mathbb{N} \rightarrow \mathbb{N}$
easy problems:

- counting problem $F : \{0, 1\}^* \times \mathbb{N} \to \mathbb{N}$
- fixed parameter tractable: can compute $F(x, k)$ in time $g(k)|x|^c$ for a computable function $g$ and constant $c$
easy problems:

- counting problem $F : \{0, 1\}^* \times \mathbb{N} \rightarrow \mathbb{N}$
- fixed parameter tractable: can compute $F(x, k)$ in time $g(k)|x|^c$ for a computable function $g$ and constant $c$
- example: counting vertex covers of size $k$
Fixed parameter tractability?

easy problems:

- counting problem $F : \{0, 1\}^* \times \mathbb{N} \rightarrow \mathbb{N}$
- fixed parameter tractable: can compute $F(x, k)$ in time $g(k)|x|^c$ for a computable function $g$ and constant $c$
- example: counting vertex covers of size $k$

hard problems:
Fixed parameter tractability?

easy problems:

- counting problem $F : \{0, 1\}^* \times \mathbb{N} \to \mathbb{N}$
- fixed parameter tractable: can compute $F(x, k)$ in time $g(k) |x|^c$ for a computable function $g$ and constant $c$
- example: counting vertex covers of size $k$

hard problems:

- $F$ is $\#W[1]$-hard, if counting $k$-cliques reduces to $F$
Fixed parameter tractability?

easy problems:

- counting problem $F : \{0, 1\}^* \times \mathbb{N} \rightarrow \mathbb{N}$
- fixed parameter tractable: can compute $F(x, k)$ in time $g(k)|x|^c$ for a computable function $g$ and constant $c$
- example: counting vertex covers of size $k$

hard problems:

- $F$ is #W[1]-hard, if counting $k$-cliques reduces to $F$
- conjecture: #W[1]-hard problems are not fixed parameter tractable
Bad news

**Theorem**

#ACQ is #W[1]-hard for stars parameterized by the number of leaves.
Theorem

#ACQ is #W[1]-hard for stars parameterized by the number of leaves.

- quantified star size of the hard instances is the number of leaves
Theorem

\#ACQ is \#W[1]-hard for stars parameterized by the number of leaves.

- quantified star size of the hard instances is the number of leaves
- consequence: \#ACQ not fixed parameter tractable w.r.t. quantified star size
Bigger tractable classes of \#ACQ?

- S-hypergraph: hypergraph $H$ with vertex subset $S$ (free variables)
Bigger tractable classes of #ACQ?

- S-hypergraph: hypergraph $H$ with vertex subset $S$ (free variables)
- #ACQ tractable for class $C$ of $S$-hypergraphs: for every conjunctive query with
Bigger tractable classes of \#ACQ?

- S-hypergraph: hypergraph $H$ with vertex subset $S$ (free variables)
- \#ACQ tractable for class $\mathcal{C}$ of $S$-hypergraphs: for every conjunctive query with
  - hypergraph $H$
Bigger tractable classes of \#ACQ?

- \(S\)-hypergraph: hypergraph \(H\) with vertex subset \(S\) (free variables)
- \#ACQ tractable for class \(C\) of \(S\)-hypergraphs: for every conjunctive query with
  - hypergraph \(H\)
  - free variable set \(S\)
Bigger tractable classes of #ACQ?

- S-hypergraph: hypergraph $H$ with vertex subset $S$ (free variables)
- #ACQ tractable for class $C$ of $S$-hypergraphs: for every conjunctive query with
  - hypergraph $H$
  - free variable set $S$
  - such that $(H, S) \in C$
S-hypergraph: hypergraph $H$ with vertex subset $S$ (free variables)

#ACQ tractable for class $C$ of $S$-hypergraphs: for every conjunctive query with

- hypergraph $H$
- free variable set $S$
- such that $(H, S) \in C$

we can count the answers efficiently
Bigger tractable classes of \#ACQ?

- **S-hypergraph**: hypergraph $H$ with vertex subset $S$ (free variables)

- \#ACQ tractable for class $C$ of $S$-hypergraphs: for every conjunctive query with
  - hypergraph $H$
  - free variable set $S$
  - such that $(H, S) \in C$

  we can count the answers efficiently

- Theorem from before: Let $C$ be the class of acyclic $S$-hypergraphs of quantified starsize $k$. Then \#ACQ is tractable for $C$
Bigger tractable classes of \#ACQ?

- **S-hypergraph**: hypergraph $H$ with vertex subset $S$ (free variables)
- \#ACQ tractable for class $C$ of $S$-hypergraphs: for every conjunctive query with
  - hypergraph $H$
  - free variable set $S$
  - such that $(H, S) \in C$

  we can count the answers efficiently

- Theorem from before: Let $C$ be the class of acyclic $S$-hypergraphs of quantified starsize $k$. Then \#ACQ is tractable for $C$

- reverse direction also true!
Bounded star size is necessary

**Theorem**

Let $C$ be a recursively enumerable class of acyclic $S$-hypergraphs. If $\#ACQ$ is tractable for $C$, then $C$ is of bounded quantified star size (assuming $\text{FPT} \neq \#\text{W}[1]$).
Theorem

Let $C$ be a recursively enumerable class of acyclic $S$-hypergraphs. If $\#\text{ACQ}$ is tractable for $C$, then $C$ is of bounded quantified star size (assuming $\text{FPT} \neq \#\text{W}[1]$).

- recursive enumerability just a minor technical restriction
Theorem

Let $C$ be a recursively enumerable class of acyclic $S$-hypergraphs. If $\#ACQ$ is tractable for $C$, then $C$ is of bounded quantified star size (assuming $\text{FPT} \neq \#W[1]$).

- recursive enumerability just a minor technical restriction
- under reasonable assumptions quantified star size is the only restriction that makes $\#ACQ$ tractable!
have completely characterized the tractable subclasses of #ACQ by parameter quantified star size

> generalization to “nearly acyclic” queries (bounded treewidth, cliquewidth, hypertree width,...)?
Conclusion

- have completely characterized the tractable subclasses of \#ACQ by parameter quantified star size
- quantified star size is efficiently computable
Conclusion

- have completely characterized the tractable subclasses of #ACQ by parameter quantified star size
- quantified star size is efficiently computable
- everything generalizes to weighted counting
Conclusion

- have completely characterized the tractable subclasses of \#ACQ by parameter quantified star size
- quantified star size is efficiently computable
- everything generalizes to weighted counting
- generalization to “nearly acyclic” queries (bounded treewidth, cliquewidth, hypertree width,...)?
have completely characterized the tractable subclasses of \#ACQ by parameter quantified star size
- quantified star size is efficiently computable
- everything generalizes to weighted counting
- generalization to “nearly acyclic” queries (bounded treewidth, cliquewidth, hypertree width,...)?
- classes of \#ACQ that allow fixed parameter counting?
Thank you for your attention!