Query Evaluation with Model Counting and Knowledge Compilation: A Survey of Old and Recent Results

Antoine Amarilli

January 16th, 2024

Télécom Paris

Table of contents

Explanations for Boolean queries

- We focus on **query evaluation**: evaluate a **query** on **data**

$$Q(x, z) : \exists y \; R(x, y) \wedge S(y, z)$$

Query

R	
a	b
a	b'

S	
b	c
b'	c'

Instance

x	z
a	c
a	c'

Results

Explanations for Boolean queries

- We focus on **query evaluation**: evaluate a **query** on **data**

$$Q(x, z) : \exists y \; R(x, y) \wedge S(y, z)$$

Query

R	
a	b
a	b'

S	
b	c
b'	c'

Instance

x	z
a	c
a	c'

Results

- We can test all answers (polynomial for fixed arity): focus on **Boolean queries**

Explanations for Boolean queries

- We focus on **query evaluation**: evaluate a **query** on **data**

$Q : \exists xyz\ R(x, y) \wedge S(y, z)$

Query

R	
a	b
a	b'

S	
b	c
b'	c'

Instance

$\longrightarrow$ YES / NO

Answer

- We can test all answers (polynomial for fixed arity): focus on **Boolean queries**

Explanations for Boolean queries

- We focus on **query evaluation**: evaluate a **query** on **data**

$Q : \exists xyz\ R(x,y) \wedge S(y,z)$

Query

R	
a	b
a	b'

S	
b	c
b'	c'

Instance

YES / NO

Answer

- We can test all answers (polynomial for fixed arity): focus on **Boolean queries**
- We want to **explain** the Boolean result by considering all **subsets** of input facts

Explanations for Boolean queries

- We focus on query evaluation: evaluate a query on data

$$Q : \exists xyz\ R(x, y) \land S(y, z)$$

Query

R	
a	***b***
a	*b′*

S	
b	*c*
b′	***c′***

Instance

YES / **NO**

Answer

- We can test all answers (polynomial for fixed arity): focus on Boolean queries
- We want to explain the Boolean result by considering all subsets of input facts

Explanations for Boolean queries

- We focus on query evaluation: evaluate a query on data

$Q : \exists xyz\ R(x, y) \wedge S(y, z)$

Query

R

a	***b***
a	b'

S

b	***c***
b'	c'

Instance

YES / NO

Answer

- We can test all answers (polynomial for fixed arity): focus on Boolean queries
- We want to explain the Boolean result by considering all subsets of input facts

Model counting and knowledge compilation

Query: $Q : \exists xyz\ R(x, y) \land S(y, z)$

R	
a	b
a	b'

S	
b	c
b'	c'

Model counting and knowledge compilation

Query: $Q : \exists xyz\; R(x, y) \wedge S(y, z)$

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

- Add a unique identifier to every fact of the instance I

Model counting and knowledge compilation

Query: $Q : \exists xyz\ R(x,y) \land S(y,z)$

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

- Add a **unique identifier** to every fact of the instance I
- There are **exponentially many** subsets of I: "possible worlds"

Model counting and knowledge compilation

Query: $Q : \exists xyz\ R(x,y) \wedge S(y,z)$

R

a	b	r_1
a	b'	r_2

S

b	c	s_1
b'	c'	s_2

- Add a unique identifier to every fact of the instance I
- There are exponentially many subsets of I: "possible worlds"
- We want to understand on which possible worlds the query Q is true...

Model counting and knowledge compilation

Query: $Q : \exists xyz\ R(x,y) \wedge S(y,z)$

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

- Add a **unique identifier** to every fact of the instance I
- There are **exponentially many** subsets of I: "possible worlds"
- We want to **understand** on which possible worlds the query Q is true...
  - **Model counting**: count **how many possible worlds** satisfy Q (here: 7)
    - **Uniform reliability** (UR): unweighted counting
    - **Probabilistic query evaluation** (PQE): add probabilities (weights) to each fact
    - **Shapley values**: another kind of aggregate

Model counting and knowledge compilation

Query: $Q : \exists xyz\, R(x, y) \wedge S(y, z)$

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

- Add a **unique identifier** to every fact of the instance I
- There are **exponentially many** subsets of I: "possible worlds"
- We want to **understand** on which possible worlds the query Q is true...
  - **Model counting**: count **how many possible worlds** satisfy Q (here: 7)
    - **Uniform reliability** (UR): unweighted counting
    - **Probabilistic query evaluation** (PQE): add probabilities (weights) to each fact
    - **Shapley values**: another kind of aggregate
  - **Knowledge compilation**: compute a **representation** of the **Boolean provenance**, i.e., the set of possible worlds satisfying Q (here: $(r_1 \wedge s_1) \vee (r_2 \wedge s_2)$)
    - Depends on the **class of representations**: circuits, diagrams, d-SDNNFs...
    - **Intensional approach**: we can use knowledge compilation for model counting

Roadmap

Goal of this talk: give an overview of results on query evaluation for:

- **Model counting:** (approximate) counting algorithms / hardness bounds
- **Knowledge compilation:** algorithms to compute circuits / circuit size lower bounds

Roadmap

Goal of this talk: give an overview of results on query evaluation for:

- **Model counting:** (approximate) counting algorithms / hardness bounds
- **Knowledge compilation:** algorithms to compute circuits / circuit size lower bounds

Settings covered in each case:

Roadmap

Goal of this talk: give an overview of results on query evaluation for:

- **Model counting:** (approximate) counting algorithms / hardness bounds
- **Knowledge compilation:** algorithms to compute circuits / circuit size lower bounds

Settings covered in each case:

- **Data complexity** of various queries on **arbitrary data**
  - $\rightarrow$ Self-join-free CQs, UCQs, homomorphism-closed queries, etc.

Roadmap

Goal of this talk: give an overview of results on query evaluation for:

- **Model counting:** (approximate) counting algorithms / hardness bounds
- **Knowledge compilation:** algorithms to compute circuits / circuit size lower bounds

Settings covered in each case:

- **Data complexity** of various queries on **arbitrary data**
  - $\rightarrow$ Self-join-free CQs, UCQs, homomorphism-closed queries, etc.
- **Data complexity** of expressive queries (MSO) on **restricted data** (treelike)

Roadmap

Goal of this talk: give an overview of results on query evaluation for:

- **Model counting:** (approximate) counting algorithms / hardness bounds
- **Knowledge compilation:** algorithms to compute circuits / circuit size lower bounds

Settings covered in each case:

- **Data complexity** of various queries on **arbitrary data**
  $\rightarrow$ Self-join-free CQs, UCQs, homomorphism-closed queries, etc.
- **Data complexity** of expressive queries (MSO) on **restricted data** (treelike)
- **Combined complexity**

Table of contents

Model counting and data complexity

- **Fix:** a Boolean query Q (maps every database to true or false)
- **Input:** a relational database I
  - For PQE: also give a **probability** to each fact in the input

Model counting and data complexity

- **Fix:** a Boolean query Q (maps every database to true or false)
- **Input:** a relational database I
  - For PQE: also give a **probability** to each fact in the input
- **Output**:
  - For UR: the **number of subsets** of I such that Q is satisfied
  - For PQE: the **probability** of getting such a subset (assuming independence)

Model counting and data complexity

- **Fix:** a Boolean query Q (maps every database to true or false)
- **Input:** a relational database I
  - For PQE: also give a **probability** to each fact in the input
- **Output**:
  - For UR: the **number of subsets** of I such that Q is satisfied
  - For PQE: the **probability** of getting such a subset (assuming independence)
- **Data complexity:** study complexity of $\mathrm{UR}(Q)$ and $\mathrm{PQE}(Q)$ as a **function of I**

Model counting and data complexity

- **Fix:** a Boolean query Q (maps every database to true or false)
- **Input:** a relational database I
  - For PQE: also give a **probability** to each fact in the input
- **Output**:
  - For UR: the **number of subsets** of I such that Q is satisfied
  - For PQE: the **probability** of getting such a subset (assuming independence)
- **Data complexity:** study complexity of $\mathrm{UR}(Q)$ and $\mathrm{PQE}(Q)$ as a **function of I**

$\rightarrow$ **For which queries Q** are these problems $\mathrm{UR}(Q)$ and $\mathrm{PQE}(Q)$ solvable in PTIME?

Data complexity of model counting for SJFCQs

A **self-join free conjunctive query** (SJFCQ) is a CQ without repeated relations

Theorem (Dalvi, Suciu, VLDB'04, VLDBJ'07)

The following dichotomy holds on SJFCQs Q:

- *If Q is **hierarchical**, then $\mathrm{PQE}(Q)$ is in **PTIME***
- *Otherwise, $\mathrm{PQE}(Q)$ is **#P-hard***

Data complexity of model counting for SJFCQs

A self-join free conjunctive query (SJFCQ) is a CQ without repeated relations

Theorem (Dalvi, Suciu, VLDB'04, VLDBJ'07)

The following dichotomy holds on SJFCQs Q:

- *If Q is hierarchical, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

Theorem (A., Kimelfeld, ICDT'21, LMCS'22)

The same dichotomy holds for uniform reliability $(\mathrm{UR}(Q))$

Data complexity of model counting for UCQs

For the class of **unions of conjunctive queries** (UCQs):

Theorem (Dalvi, Suciu, JACM'12)

The following dichotomy holds on UCQs Q:

- *If Q is **safe**, then $\mathrm{PQE}(Q)$ is in **PTIME***
- *Otherwise, $\mathrm{PQE}(Q)$ is **#P-hard***

Data complexity of model counting for UCQs

For the class of unions of conjunctive queries (UCQs):

Theorem (Dalvi, Suciu, JACM'12)

The following dichotomy holds on UCQs Q:

- *If Q is safe, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

Theorem (Kenig, Suciu, PODS'21)

- *The same dichotomy holds even when probabilities can only be $1/2$ and 1 (and 0)*
- *For some unsafe UCQs, hardness already holds for uniform reliability*

Data complexity of model counting for UCQs

For the class of unions of conjunctive queries (UCQs):

Theorem (Dalvi, Suciu, JACM'12)

The following dichotomy holds on UCQs Q:

- *If Q is safe, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

Theorem (Kenig, Suciu, PODS'21)

- *The same dichotomy holds even when probabilities can only be $1/2$ and 1 (and 0)*
- *For some unsafe UCQs, hardness already holds for uniform reliability*

Is $\mathrm{UR}(Q)$ #P-hard for every unsafe UCQ, like $\mathrm{PQE}(Q)$? Open

Data complexity of model counting for homomorphism-closed queries

- A query Q is homomorphism-closed if satisfaction is preserved by homomorphisms

Data complexity of model counting for homomorphism-closed queries

- A query Q is homomorphism-closed if satisfaction is preserved by homomorphisms
  - $\rightarrow$ If I satisfies Q and I has a homomorphism to I' then I' satisfies Q
  - $\rightarrow$ Ex.: CQs, UCQs, some recursive queries like regular path queries (RPQs), Datalog, etc.

Data complexity of model counting for homomorphism-closed queries

- A query Q is homomorphism-closed if satisfaction is preserved by homomorphisms
  - → If I satisfies Q and I has a homomorphism to I' then I' satisfies Q
  - → Ex.: CQs, UCQs, some recursive queries like regular path queries (RPQs), Datalog, etc.
- A homomorphism-closed query is unbounded if it is not equivalent to a UCQ

Data complexity of model counting for homomorphism-closed queries

- A query Q is homomorphism-closed if satisfaction is preserved by homomorphisms
  - → If I satisfies Q and I has a homomorphism to I' then I' satisfies Q
  - → Ex.: CQs, UCQs, some recursive queries like regular path queries (RPQs), Datalog, etc.
- A homomorphism-closed query is unbounded if it is not equivalent to a UCQ

Theorem (A., Ceylan, ICDT'20, LMCS'22)

On arity-two signatures, for any unbounded homomorphism-closed query Q, the PQE problem is #P-hard

Data complexity of model counting for homomorphism-closed queries

- A query Q is **homomorphism-closed** if satisfaction is preserved by homomorphisms
  - → If I satisfies Q and I has a homomorphism to I' then I' satisfies Q
  - → Ex.: **CQs**, **UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- A homomorphism-closed query is **unbounded** if it is **not** equivalent to a UCQ

Theorem (A., Ceylan, ICDT'20, LMCS'22)

On arity-two signatures, for any unbounded homomorphism-closed query Q, the PQE problem is #P-hard

Theorem (A., ICDT'23)

The same holds for the UR problem

Data complexity of model counting for homomorphism-closed queries

- A query Q is **homomorphism-closed** if satisfaction is preserved by homomorphisms
  - $\rightarrow$ If I satisfies Q and I has a homomorphism to I' then I' satisfies Q
  - $\rightarrow$ Ex.: **CQs**, **UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- A homomorphism-closed query is **unbounded** if it is **not** equivalent to a UCQ

Theorem (A., Ceylan, ICDT'20, LMCS'22)

On arity-two signatures, for any unbounded homomorphism-closed query Q, the PQE problem is #P-hard

Theorem (A., ICDT'23)

The same holds for the UR problem

Do these hardness results extend to **higher-arity signatures**? **Open**

Data complexity of model counting: Restricting the instances

Another way to make the problem tractable: restrict the input instance!

- Fix: a query Q and an instance family $\mathcal{I}$

Data complexity of model counting: Restricting the instances

Another way to make the problem tractable: **restrict the input instance**!

- **Fix:** a query Q **and an instance family** $\mathcal{I}$
- **Input:** a relational database I **in** $\mathcal{I}$
  - For PQE: also give a probability to each fact in the input

Data complexity of model counting: Restricting the instances

Another way to make the problem tractable: **restrict the input instance**!

- **Fix:** a query Q **and an instance family** $\mathcal{I}$
- **Input:** a relational database I **in** $\mathcal{I}$
  - For PQE: also give a probability to each fact in the input
- **Output**:
  - For UR: the number of subsets of I such that Q is satisfied
  - For PQE: the probability of getting such a subset (assuming independence)

Data complexity of model counting: Restricting the instances

Another way to make the problem tractable: restrict the input instance!

- Fix: a query Q and an instance family $\mathcal{I}$
- Input: a relational database I in $\mathcal{I}$
  - For PQE: also give a probability to each fact in the input
- Output:
  - For UR: the number of subsets of I such that Q is satisfied
  - For PQE: the probability of getting such a subset (assuming independence)

$\rightarrow$ For which queries Q and which instance families $\mathcal{I}$ are these problems $\mathrm{UR}(Q, \mathcal{I})$ and $\mathrm{PQE}(Q, \mathcal{I})$ solvable in PTIME?

Data complexity of model counting: MSO on treelike instances

- Treewidth is a parameter intuitively measuring how close a graph is to a tree

Data complexity of model counting: MSO on treelike instances

- Treewidth is a parameter intuitively measuring how close a graph is to a tree
- An instance family $\mathcal{I}$ is treelike if there is a constant $k \in \mathbb{N}$ such that all instances in $\mathcal{I}$ have treewidth $\leq k$

Data complexity of model counting: MSO on treelike instances

- Treewidth is a parameter intuitively measuring how close a graph is to a tree
- An instance family $\mathcal{I}$ is treelike if there is a constant $k \in \mathbb{N}$ such that all instances in $\mathcal{I}$ have treewidth $\leq k$
- Monadic second-order logic (MSO) is a query language extending first-order logic with quantification over sets

Data complexity of model counting: MSO on treelike instances

- Treewidth is a parameter intuitively measuring how close a graph is to a tree
- An instance family $\mathcal{I}$ is treelike if there is a constant $k \in \mathbb{N}$ such that all instances in $\mathcal{I}$ have treewidth $\leq k$
- Monadic second-order logic (MSO) is a query language extending first-order logic with quantification over sets

Theorem (A., Bourhis, Senellart, ICALP'15)

For any MSO query Q and treelike instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q)$ is in PTIME

Data complexity of model counting: MSO on treelike instances

- **Treewidth** is a parameter intuitively measuring how close a graph is to a tree
- An instance family $\mathcal{I}$ is **treelike** if there is a constant $k \in \mathbb{N}$ such that all instances in $\mathcal{I}$ have treewidth $\leq k$
- **Monadic second-order logic** (MSO) is a query language extending first-order logic with quantification over sets

Theorem (A., Bourhis, Senellart, ICALP'15)

For any MSO query Q and treelike instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q)$ is in PTIME

- In particular: PQE is in PTIME for **tree automata** over **probabilistic trees** (already in: Cohen, Kimelfeld, Sagiv, PODS'09)

Data complexity of model counting: MSO on treelike instances

- Treewidth is a parameter intuitively measuring how close a graph is to a tree
- An instance family $\mathcal{I}$ is treelike if there is a constant $k \in \mathbb{N}$ such that all instances in $\mathcal{I}$ have treewidth $\leq k$
- Monadic second-order logic (MSO) is a query language extending first-order logic with quantification over sets

Theorem (A., Bourhis, Senellart, ICALP'15)

For any MSO query Q and treelike instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q)$ is in PTIME

- In particular: PQE is in PTIME for tree automata over probabilistic trees (already in: Cohen, Kimelfeld, Sagiv, PODS'09)

$\rightarrow$ Is treelikeness the only condition on the instance that makes PQE tractable?

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  - $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  - $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$
- A non-treelike instance family $\mathcal{I}$ is treewidth-constructible if, given $k \in \mathbb{N}$, we can compute in time Poly(k) an instance in $\mathcal{I}$ with treewidth at least k

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  - $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$
- A non-treelike instance family $\mathcal{I}$ is treewidth-constructible if, given $k \in \mathbb{N}$, we can compute in time Poly(k) an instance in $\mathcal{I}$ with treewidth at least k

Theorem (A., Monet, MFCS'22)

For any treewidth-constructible instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q_2)$ is #P-hard under ZPP reductions

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$
- A non-treelike instance family $\mathcal{I}$ is **treewidth-constructible** if, given $k \in \mathbb{N}$, we can compute in time Poly(k) an instance in $\mathcal{I}$ with treewidth at least k

Theorem (A., Monet, MFCS'22)

For any treewidth-constructible instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q_2)$ is #P-hard under ZPP reductions

(Uses polynomial bounds on grid minor extraction (Chekuri, Chuzhoy, JACM'16))

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$
- A non-treelike instance family $\mathcal{I}$ is treewidth-constructible if, given $k \in \mathbb{N}$, we can compute in time Poly(k) an instance in $\mathcal{I}$ with treewidth at least k

Theorem (A., Monet, MFCS'22)

For any treewidth-constructible instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q_2)$ is #P-hard under ZPP reductions

(Uses polynomial bounds on grid minor extraction (Chekuri, Chuzhoy, JACM'16))

- Does this intractability result also hold for UR? Open

Data complexity: Counting matchings on unbounded-treewidth graphs

- Fix a binary relation R, let Q_2 be the query asking if there are two incident facts
  $\rightarrow$ Q_2 is a UCQ with inequalities: $(R(x,y) \vee R(y,x)) \wedge (R(y,z) \vee R(z,y)) \wedge x \neq z$
- A non-treelike instance family $\mathcal{I}$ is **treewidth-constructible** if, given $k \in \mathbb{N}$, we can compute in time $\text{Poly}(k)$ an instance in $\mathcal{I}$ with treewidth at least k

Theorem (A., Monet, MFCS'22)

For any treewidth-constructible instance family $\mathcal{I}$, the problem $\mathrm{PQE}(Q_2)$ is #P-hard under ZPP reductions

(Uses polynomial bounds on grid minor extraction (Chekuri, Chuzhoy, JACM'16))

- Does this intractability result also hold for UR? Open
- For which other queries than Q_2 does this hold? What about higher arity? Open

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

This is intractable already without probabilities so we restrict the instance and query

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

This is intractable already without probabilities so we restrict the instance and query

$\xrightarrow{R} \xrightarrow{S} \xrightarrow{T}$

1WP: one-way paths

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

This is intractable already without probabilities so we restrict the instance and query

$\xrightarrow{R} \xrightarrow{S} \xrightarrow{T}$

1WP: one-way paths

$\xrightarrow{R} \xleftarrow{R} \xleftarrow{T}$

2WP: two-way paths

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

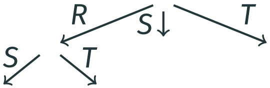
This is intractable already without probabilities so we restrict the instance and query

$\xrightarrow{R}\ \xrightarrow{S}\ \xrightarrow{T}$

1WP: one-way paths

$\xrightarrow{R}\ \xleftarrow{R}\ \xleftarrow{T}$

2WP: two-way paths



DWT: downward trees

Model counting: Combined complexity

- Input: an arity-two signature, a query Q, an instance I with probabilities on facts
- Output: the probability that Q is true in I

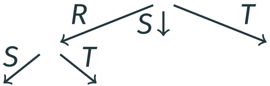
This is intractable already without probabilities so we restrict the instance and query

$\xrightarrow{R}\ \xrightarrow{S}\ \xrightarrow{T}$

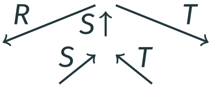
1WP: one-way paths

$\xrightarrow{R}\ \xleftarrow{R}\ \xleftarrow{T}$

2WP: two-way paths



DWT: downward trees



PT: polytrees

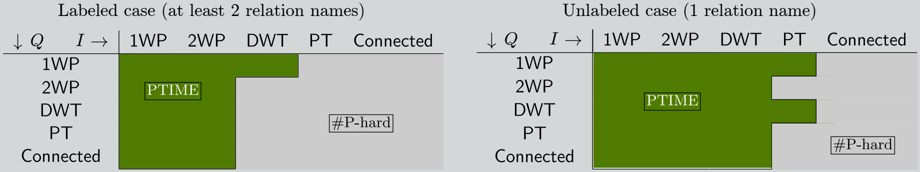
Model counting: Combined complexity

- Input: an **arity-two** signature, a **query** Q, an **instance** I with **probabilities** on facts
- Output: the **probability** that Q is true in I

This is **intractable** already without probabilities so we restrict the **instance** and **query**

1WP: one-way paths 2WP: two-way paths DWT: downward trees PT: polytrees

Theorem (A., Monet, Senellart, PODS'17)



Model counting: Combined complexity

- Input: an **arity-two** signature, a **query** Q, an **instance** I with **probabilities** on facts
- Output: the **probability** that Q is true in I

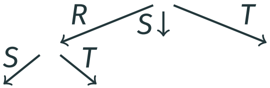
This is **intractable** already without probabilities so we restrict the **instance** and **query**

$\xrightarrow{R}\ \xrightarrow{S}\ \xrightarrow{T}$

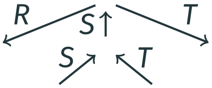
1WP: one-way paths

$\xrightarrow{R}\ \xleftarrow{R}\ \xleftarrow{T}$

2WP: two-way paths



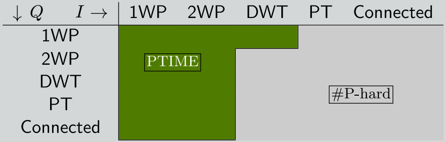
DWT: downward trees



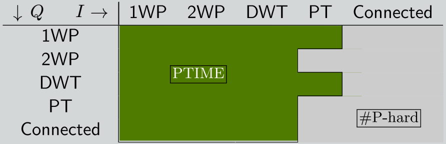
PT: polytrees

Theorem (A., Monet, Senellart, PODS'17)

Labeled case (at least 2 relation names)



Unlabeled case (1 relation name)



Do hardness results still hold for **UR**? **Open** (we haven't thought about it)

Model counting: Approximation

Model counting problems (UR/PQE) ask for an exact answer. Is it easier to get an approximate answer?

Model counting: Approximation

Model counting problems (UR/PQE) ask for an **exact answer**. Is it easier to get an **approximate answer**?

Goal: design an algorithm running in **PTIME** in the input and in the **error $\epsilon > 0$** which...

Model counting: Approximation

Model counting problems (UR/PQE) ask for an **exact answer**. Is it easier to get an **approximate answer**?

Goal: design an algorithm running in **PTIME** in the input and in the **error** $\epsilon > 0$ which...

Additive FPRAS:

- ... with probability $> 1/2$, gives an answer x' with **additive error** $\leq \epsilon$ wrt the true answer x

Model counting: Approximation

Model counting problems (UR/PQE) ask for an **exact answer**. Is it easier to get an **approximate answer**?

Goal: design an algorithm running in **PTIME** in the input and in the **error** $\epsilon > 0$ which...

Additive FPRAS:

- ... with probability $> 1/2$, gives an answer x' with **additive error** $\leq \epsilon$ wrt the true answer x
  - → We want, with proba $> 1/2$, that $x - \epsilon < x' < x + \epsilon$

Model counting: Approximation

Model counting problems (UR/PQE) ask for an **exact answer**. Is it easier to get an **approximate answer**?

Goal: design an algorithm running in **PTIME** in the input and in the **error** $\epsilon > 0$ which...

Additive FPRAS:

- ... with probability $> 1/2$, gives an answer x' with **additive error** $\leq \epsilon$ wrt the true answer x
  - → We want, with proba $> 1/2$, that $x - \epsilon < x' < x + \epsilon$
  - → **Always possible** (Monte Carlo) if you can **sample** and **evaluate** in PTIME

Model counting: Approximation

Model counting problems (UR/PQE) ask for an **exact answer**. Is it easier to get an **approximate answer**?

Goal: design an algorithm running in **PTIME** in the input and in the **error** $\epsilon > 0$ which...

Additive FPRAS:

- ... with probability $> 1/2$, gives an answer x' with **additive error** $\leq \epsilon$ wrt the true answer x
  - → We want, with proba $> 1/2$, that $x - \epsilon < x' < x + \epsilon$
  - → **Always possible** (Monte Carlo) if you can **sample** and **evaluate** in PTIME

FPRAS:

- ... with probability $> 1/2$, gives an answer x' with **multiplicative error** $\leq \epsilon$ wrt the true answer x
  - → We want, with proba $> 1/2$, that $(1 - \epsilon)x < x' < (1 + \epsilon)x$
  - → **Rest of the talk only covers this notion**

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For data complexity for UCQs:

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For data complexity for UCQs:
  - $\rightarrow$ Always possible, via Karp-Luby algorithm on the disjunction of minimal matches

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For data complexity for UCQs:
  $\rightarrow$ Always possible, via Karp-Luby algorithm on the disjunction of minimal matches
- For data complexity for homomorphism-closed queries:

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For data complexity for UCQs:
  - $\rightarrow$ Always possible, via Karp-Luby algorithm on the disjunction of minimal matches
- For data complexity for homomorphism-closed queries:
  - $\rightarrow$ Karp-Luby no longer applies (exponentially many minimal matches)

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For data complexity for UCQs:
  - $\rightarrow$ Always possible, via Karp-Luby algorithm on the disjunction of minimal matches
- For data complexity for homomorphism-closed queries:
  - $\rightarrow$ Karp-Luby no longer applies (exponentially many minimal matches)
  - $\rightarrow$ Open (ongoing work)

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For **data complexity** for **UCQs**:
  - $\rightarrow$ Always possible, via **Karp-Luby algorithm** on the disjunction of minimal matches
- For **data complexity** for **homomorphism-closed queries**:
  - $\rightarrow$ Karp-Luby no longer applies (exponentially many minimal matches)
  - $\rightarrow$ **Open** (ongoing work)
- For **combined complexity**, we want a **combined FPRAS**:

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For **data complexity** for **UCQs**:
  - $\rightarrow$ Always possible, via **Karp-Luby algorithm** on the disjunction of minimal matches
- For **data complexity** for **homomorphism-closed queries**:
  - $\rightarrow$ Karp-Luby no longer applies (exponentially many minimal matches)
  - $\rightarrow$ **Open** (ongoing work)
- For **combined complexity**, we want a **combined FPRAS**:
  - Input: query Q, instance I with probabilities, error $\epsilon > 0$
  - Output: with proba $> 1/2$, in polynomial time, **approximation** of the probability of Q on I up to **multiplicative** error ϵ

Model counting: Easy approximation results

When can we have an FPRAS for model counting?

- For **data complexity** for **UCQs**:
  - $\rightarrow$ Always possible, via **Karp-Luby algorithm** on the disjunction of minimal matches
- For **data complexity** for **homomorphism-closed queries**:
  - $\rightarrow$ Karp-Luby no longer applies (exponentially many minimal matches)
  - $\rightarrow$ **Open** (ongoing work)
- For **combined complexity**, we want a **combined FPRAS**:
  - Input: query Q, instance I with probabilities, error $\epsilon > 0$
  - Output: with proba $> 1/2$, in polynomial time, **approximation** of the probability of Q on I up to **multiplicative** error ϵ
  - $\rightarrow$ When can we obtain such an algorithm?

Model counting: Combined complexity of approximation

Theorem (van Bremen, Meel, PODS'23)

There is a combined FPRAS for PQE with SJFCQs Q of bounded hypertreewidth and arbitrary instances I with probabilities on facts

Model counting: Combined complexity of approximation

Theorem (van Bremen, Meel, PODS'23)

There is a combined FPRAS for PQE with SJFCQs Q *of bounded hypertreewidth and arbitrary instances* I *with probabilities on facts*

What about self-joins?

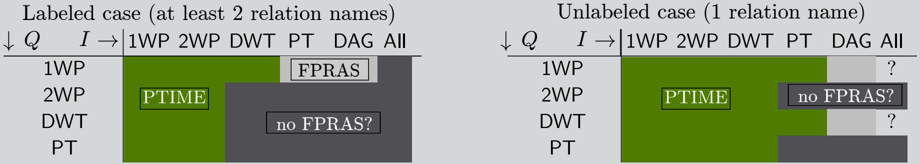
Model counting: Combined complexity of approximation

Theorem (van Bremen, Meel, PODS'23)

*There is a combined FPRAS for PQE with SJFCQs **Q** of bounded hypertreewidth and arbitrary instances **I** with probabilities on facts*

What about **self-joins**?

Theorem (A., van Bremen, Meel, ICDT'24)



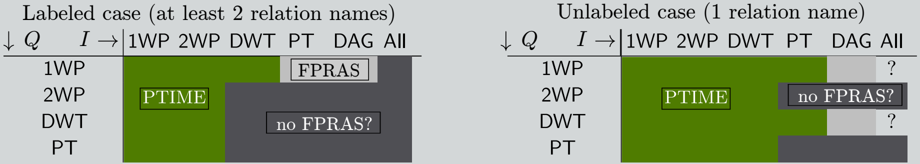
Model counting: Combined complexity of approximation

Theorem (van Bremen, Meel, PODS'23)

There is a combined FPRAS for PQE with SJFCQs **Q** *of bounded hypertreewidth and arbitrary instances* **I** *with probabilities on facts*

What about **self-joins**?

Theorem (A., van Bremen, Meel, ICDT'24)



(Using **FPRAS for #NFA** for word and tree automata (Arenas et al., JACM'21, PODS'21))

FPRAS inexistence results assume $\mathrm{RP} \neq \mathrm{NP}$

Table of contents

Boolean provenance

- Fix: a Boolean query Q
- Input: a relational database I
- Output: a representation of the Boolean provenance of Q on I

Boolean provenance

- **Fix:** a Boolean query Q
- **Input:** a relational database I
- **Output:** a **representation** of the **Boolean provenance** of Q on I

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

Query: $Q : \exists xyz\ R(x,y) \wedge S(y,z)$

Boolean provenance

- **Fix:** a Boolean query Q
- **Input:** a relational database I
- **Output:** a **representation** of the **Boolean provenance** of Q on I

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

Query: $Q : \exists xyz\ R(x,y) \wedge S(y,z)$

Boolean provenance: $\Phi : (r_1 \wedge s_1) \vee (r_2 \wedge s_2)$

Boolean provenance

- **Fix:** a Boolean query Q
- **Input:** a relational database I
- **Output:** a **representation** of the **Boolean provenance** of Q on I

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

Query: $Q : \exists xyz\, R(x,y) \wedge S(y,z)$

Boolean provenance: $\Phi : (r_1 \wedge s_1) \vee (r_2 \wedge s_2)$

How is the provenance Φ related to **model counting**?

- The **satisfying assignments** of Φ are the **subinstances of I where Q holds**

Boolean provenance

- Fix: a Boolean query Q
- Input: a relational database I
- Output: a representation of the Boolean provenance of Q on I

	R	
a	b	r_1
a	b'	r_2

	S	
b	c	s_1
b'	c'	s_2

Query: $Q : \exists xyz\ R(x, y) \wedge S(y, z)$

Boolean provenance: $\Phi : (r_1 \wedge s_1) \vee (r_2 \wedge s_2)$

How is the provenance Φ related to model counting?

- The satisfying assignments of Φ are the subinstances of I where Q holds
- UR asks how many satisfying assignments Φ has

Boolean provenance

- **Fix:** a Boolean query Q
- **Input:** a relational database I
- **Output:** a **representation** of the **Boolean provenance** of Q on I

R		
a	b	r_1
a	b'	r_2

S		
b	c	s_1
b'	c'	s_2

Query: $Q : \exists xyz\ R(x,y) \wedge S(y,z)$

Boolean provenance: $\Phi : (r_1 \wedge s_1) \vee (r_2 \wedge s_2)$

How is the provenance Φ related to **model counting**?

- The **satisfying assignments** of Φ are the **subinstances of I where Q holds**
- UR asks **how many satisfying assignments** Φ has
- PQE asks for the **probability** that Φ evaluates to true

Easy results on Boolean provenance

We can tractably compute Boolean provenance in data complexity:

Theorem (folklore)

For any fixed UCQ Q, given an instance I, we can compute the provenance of Q on I as a Boolean formula in PTIME data complexity

Easy results on Boolean provenance

We can tractably compute Boolean provenance in data complexity:

Theorem (folklore)

For any fixed UCQ Q, given an instance I, we can compute the provenance of Q on I as a Boolean formula in PTIME data complexity

Proof: Simply test all possible variable assignments (disjunctive normal form formula)

Easy results on Boolean provenance

We can tractably compute Boolean provenance in **data complexity**:

Theorem (folklore)

For any fixed UCQ Q*, given an instance* I*, we can compute the provenance of* Q *on* I *as a Boolean formula in PTIME data complexity*

Proof: Simply test **all possible variable assignments** (disjunctive normal form formula)

Theorem (Deutch et al., ICDT'14)

For any fixed Datalog program Q*, given an instance* I*, we can compute the provenance of* Q *on* I *in PTIME data complexity as a Boolean circuit*

Easy results on Boolean provenance

We can tractably compute Boolean provenance in **data complexity**:

Theorem (folklore)

For any fixed UCQ Q, given an instance I, we can compute the provenance of Q on I as a Boolean formula in PTIME data complexity

Proof: Simply test **all possible variable assignments** (disjunctive normal form formula)

Theorem (Deutch et al., ICDT'14)

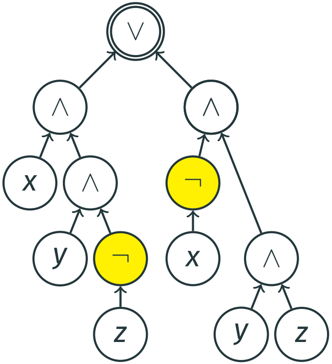
For any fixed Datalog program Q, given an instance I, we can compute the provenance of Q on I in PTIME data complexity as a Boolean circuit

Goal: provenance representation in **tractable circuit classes** from knowledge compilation

Tractable circuits: d-DNNF and d-SDNNF

Tractable circuit class: **d-DNNF:**

- $\neg$ are only applied to variables

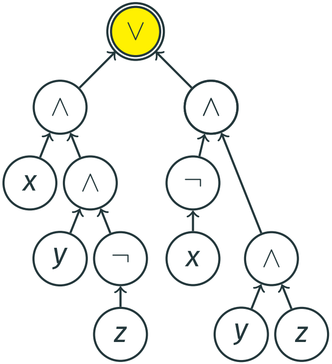


Tractable circuits: d-DNNF and d-SDNNF

Tractable circuit class: **d-DNNF:**

- $\neg$ are only applied to variables
- $\vee$ are all deterministic:

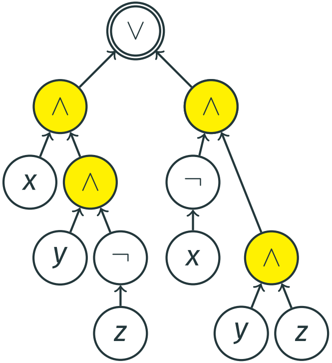
The inputs are mutually exclusive
(= no valuation makes two inputs simultaneously evaluate to 1)



Tractable circuits: d-DNNF and d-SDNNF

Tractable circuit class: **d-DNNF:**

- $\neg$ are only applied to variables
- $\vee$ are all **deterministic**:
The inputs are **mutually exclusive**
(= no valuation makes two inputs simultaneously evaluate to 1)
- $\wedge$ are all **decomposable**:
The inputs are **independent**
(= no variable x has a path to two different inputs)



Tractable circuits: d-DNNF and d-SDNNF

Tractable circuit class: **d-DNNF:**

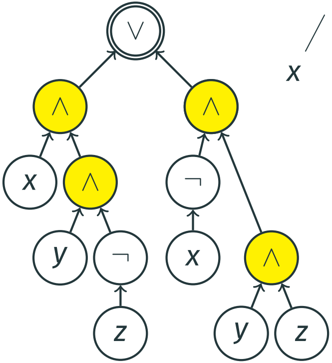
- ¬ are only applied to variables
- ∨ are all **deterministic**:

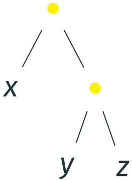
The inputs are **mutually exclusive**
(= no valuation makes two inputs simultaneously evaluate to 1)

- ∧ are all **decomposable**:

The inputs are **independent**
(= no variable x has a path to two different inputs)

Frequent extra requirement: **structuredness** (following a **vtree**), aka **d-SDNNF**





Solving PQE with d-DNNF

d-DNNF requirements...

Solving PQE with d-DNNF

d-DNNF requirements...

- $\neg$ gates only have **variables** as inputs

Solving PQE with d-DNNF

d-DNNF requirements...

- $\neg$ gates only have **variables** as inputs
- $\lor$ gates always have **mutually exclusive** inputs

Solving PQE with d-DNNF

d-DNNF requirements...

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

Solving PQE with d-DNNF

d-DNNF requirements...

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

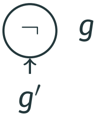
... make probability computation **easy**!

Solving PQE with d-DNNF

d-DNNF requirements…

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



Solving PQE with d-DNNF

d-DNNF requirements…

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!

$$g' \rightarrow \neg \; g$$

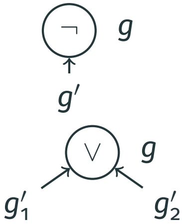
$$P(g) := 1 - P(g')$$

Solving PQE with d-DNNF

d-DNNF requirements…

- $\neg$ gates only have **variables** as inputs
- $\lor$ gates always have **mutually exclusive** inputs
- $\land$ gates are all on **independent** inputs

… make probability computation **easy**!

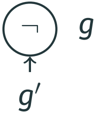


$$P(g) := 1 - P(g')$$

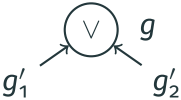
Solving PQE with d-DNNF

d-DNNF requirements… … make probability computation easy!

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs



$$P(g) := 1 - P(g')$$



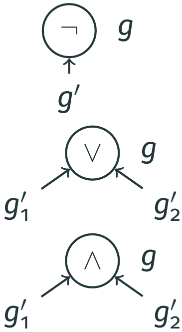
$$P(g) := P(g'_1) + P(g'_2)$$

Solving PQE with d-DNNF

d-DNNF requirements…

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



$$P(g) := 1 - P(g')$$

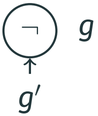
$$P(g) := P(g'_1) + P(g'_2)$$

Solving PQE with d-DNNF

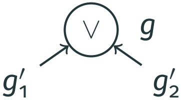
d-DNNF requirements…

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

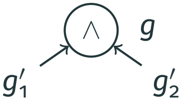
… make probability computation **easy**!



$$P(g) := 1 - P(g')$$



$$P(g) := P(g'_1) + P(g'_2)$$

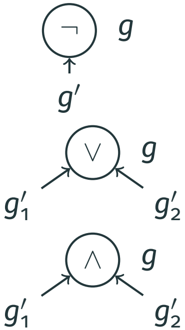


$$P(g) := P(g'_1) \times P(g'_2)$$

Solving PQE with d-DNNF

d-DNNF requirements… … make probability computation easy!

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs



$$P(g) := 1 - P(g')$$

$$P(g) := P(g'_1) + P(g'_2)$$

$$P(g) := P(g'_1) \times P(g'_2)$$

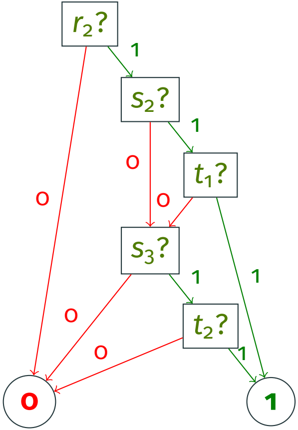
$\rightarrow$ If you can build a d-DNNF provenance representation in PTIME, then PQE is in PTIME

Other tractable circuit classes

- **Read-once formula:** Boolean formula where each variable occurs at most once
  → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation

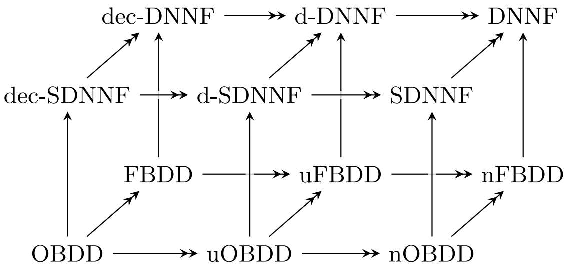
Other tractable circuit classes

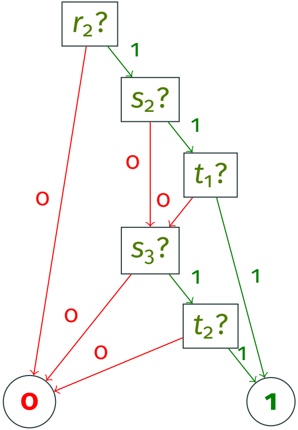
- Read-once formula: Boolean formula where each variable occurs at most once
  - $\rightarrow$ If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation
- Binary decision diagram, e.g., OBDDs



Other tractable circuit classes

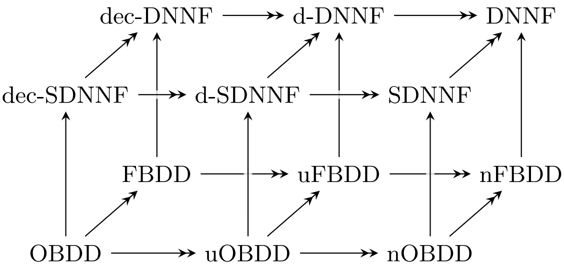
- **Read-once formula:** Boolean formula where each variable occurs at most once
  - $\rightarrow$ If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation
- **Binary decision diagram**, e.g., **OBDDs**

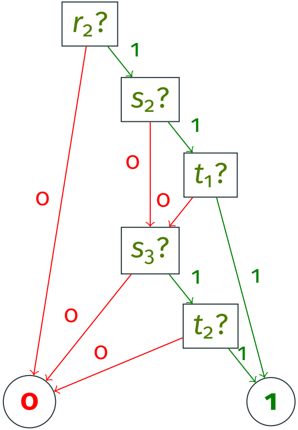




- **Read-once formula:** Boolean formula where each variable occurs at most once
  - → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation

- **Binary decision diagram**, e.g., **OBDDs**





- Generalize **d-DNNFs** to **d-Ds**: allow arbitrary negations

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): “PQE is in **PTIME**”
- **Knowledge compilation (KC)**: ”We can **tractably compute circuits** in a given class” (e.g., d-SDNNFs)

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): “PQE is in **PTIME**”
- **Knowledge compilation (KC)**: ”We can **tractably compute circuits** in a given class” (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): "PQE is in **PTIME**"
- **Knowledge compilation (KC)**: "We can **tractably compute circuits** in a given class" (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël's talk)

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): "PQE is in **PTIME**"
- **Knowledge compilation (KC)**: "We can **tractably compute circuits** in a given class" (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël's talk)

Two kinds of **intractability results:**

- $\overline{\text{MC}}$: PQE is **#P-hard**
- $\overline{\text{KC}}$: there are **no small circuits** representing provenance in a given class

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): “PQE is in **PTIME**”
- **Knowledge compilation (KC)**: ”We can **tractably compute circuits** in a given class” (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël’s talk)

Two kinds of **intractability results:**

- $\overline{\text{MC}}$: PQE is **#P-hard**
- $\overline{\text{KC}}$: there are **no small circuits** representing provenance in a given class
- $\overline{\text{MC}}$ is **conditional** (FP $\neq$ #P) but $\overline{\text{KC}}$ can be **unconditional**

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): “PQE is in **PTIME**”
- **Knowledge compilation (KC)**: ”We can **tractably compute circuits** in a given class” (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël’s talk)

Two kinds of **intractability results:**

- $\overline{\text{MC}}$**:** PQE is **#P-hard**
- $\overline{\text{KC}}$**:** there are **no small circuits** representing provenance in a given class
- $\overline{\text{MC}}$ is **conditional** (FP $\neq$ #P) but $\overline{\text{KC}}$ can be **unconditional**
- A priori **incomparable**:

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): “PQE is in **PTIME**”
- **Knowledge compilation (KC)**: ”We can **tractably compute circuits** in a given class” (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël’s talk)

Two kinds of **intractability results:**

- $\overline{\text{MC}}$**:** PQE is **#P-hard**
- $\overline{\text{KC}}$**:** there are **no small circuits** representing provenance in a given class
- $\overline{\text{MC}}$ is **conditional** (FP $\neq$ #P) but $\overline{\text{KC}}$ can be **unconditional**
- A priori **incomparable**:
  - $\overline{\text{MC}} \not\Rightarrow \overline{\text{KC}}$: we could have small circuits that are **hard to compute**

Model counting results vs knowledge compilation results

Two kinds of **tractability results:**

- **Model counting** (MC): "PQE is in **PTIME**"
- **Knowledge compilation (KC)**: "We can **tractably compute circuits** in a given class" (e.g., d-SDNNFs)
  - $\rightarrow$ We **always** have KC $\Rightarrow$ MC
  - $\rightarrow$ **Open** if MC $\Rightarrow$ KC: **intensional-extensional conjecture** (Mikaël's talk)

Two kinds of **intractability results:**

- $\overline{\text{MC}}$**:** PQE is **#P-hard**
- $\overline{\text{KC}}$**:** there are **no small circuits** representing provenance in a given class
- $\overline{\text{MC}}$ is **conditional** (FP $\neq$ #P) but $\overline{\text{KC}}$ can be **unconditional**
- A priori **incomparable**:
  - $\overline{\text{MC}} \not\Rightarrow \overline{\text{KC}}$: we could have small circuits that are **hard to compute**
  - $\overline{\text{KC}} \not\Rightarrow \overline{\text{MC}}$: PQE may be solvable **without circuits** (if int.-ext. conjecture fails)

Data complexity of provenance computation for UCQs and hom-closed queries

- For UCQs, results in (Jha, Suciu, ICDT'11, TCS'13):
  - Characterization of the UCQs for which we can compute read-once provenance

Data complexity of provenance computation for UCQs and hom-closed queries

- For UCQs, results in (Jha, Suciu, ICDT'11, TCS'13):
  - Characterization of the UCQs for which we can compute read-once provenance
  - Characterization of the UCQs for which we can compute OBDD provenance

Data complexity of provenance computation for UCQs and hom-closed queries

- For UCQs, results in (Jha, Suciu, ICDT'11, TCS'13):
  - Characterization of the UCQs for which we can compute read-once provenance
  - Characterization of the UCQs for which we can compute OBDD provenance
  - Sufficient conditions to have FBDDs and d-DNNFs

Data complexity of provenance computation for UCQs and hom-closed queries

- For UCQs, results in (Jha, Suciu, ICDT'11, TCS'13):
  - Characterization of the UCQs for which we can compute read-once provenance
  - Characterization of the UCQs for which we can compute OBDD provenance
  - Sufficient conditions to have FBDDs and d-DNNFs

- In particular, for UCQs:
  - Open if safe UCQs have small d-D provenance circuits (intensional-extensional conjecture)
  - Open if unsafe UCQs do not have small tractable circuits (beyond RO and OBDDs)

Data complexity of provenance computation for UCQs and hom-closed queries

- For UCQs, results in (Jha, Suciu, ICDT'11, TCS'13):
    - Characterization of the UCQs for which we can compute read-once provenance
    - Characterization of the UCQs for which we can compute OBDD provenance
    - Sufficient conditions to have FBDDs and d-DNNFs
- In particular, for UCQs:
    - Open if safe UCQs have small d-D provenance circuits (intensional-extensional conjecture)
    - Open if unsafe UCQs do not have small tractable circuits (beyond RO and OBDDs)
- For homomorphism-closed queries: open

Data complexity of provenance computation for treelike data

Going back to the setting of restricted instance classes, we have:

Theorem (A., Bourhis, Senellart, ICALP'15, PODS'16)

For any fixed MSO query Q *and treelike instance family* $\mathcal{I}$*, given an instance* I *in* $\mathcal{I}$*, we can compute the provenance of* Q *on* I *in PTIME as a d-SDNNF circuit*

Data complexity of provenance computation for treelike data

Going back to the setting of restricted instance classes, we have:

Theorem (A., Bourhis, Senellart, ICALP'15, PODS'16)

For any fixed MSO query $\mathbf{Q}$ *and treelike instance family* $\mathcal{I}$*, given an instance* $\mathbf{I}$ *in* $\mathcal{I}$*, we can compute the provenance of* $\mathbf{Q}$ *on* $\mathbf{I}$ *in PTIME as a d-SDNNF circuit*

Can we get a weaker representation (uOBDD, etc.)? **Open**

Data complexity of provenance computation for treelike data

Going back to the setting of restricted instance classes, we have:

Theorem (A., Bourhis, Senellart, ICALP'15, PODS'16)

For any fixed MSO query $\mathbf{Q}$ *and treelike instance family* $\mathcal{I}$*, given an instance* $\mathbf{I}$ *in* $\mathcal{I}$*, we can compute the provenance of* $\mathbf{Q}$ *on* $\mathbf{I}$ *in PTIME as a d-SDNNF circuit*

Can we get a weaker representation (uOBDD, etc.)? **Open**

What happens on high-treewidth instances?

Data complexity of provenance computation for treelike data

Going back to the setting of restricted instance classes, we have:

Theorem (A., Bourhis, Senellart, ICALP'15, PODS'16)

For any fixed MSO query Q and treelike instance family $\mathcal{I}$, given an instance I in $\mathcal{I}$, we can compute the provenance of Q on I in PTIME as a d-SDNNF circuit

Can we get a weaker representation (uOBDD, etc.)? **Open**

What happens on high-treewidth instances?

Theorem (A., Bourhis, Senellart, PODS'16; A., Monet, Senellart, ICDT'18)

For the "two incident facts" query Q_2, given an instance I, any d-SDNNF representation of the provenance of Q_2 on I is exponential: in $\Omega\left(2^{tw(I)^{1/d}}\right)$ for some $d \geq 1$

For which other queries is this true? Mostly open (some results for connected $UCQ^{\neq}$)

Data complexity of provenance computation for treelike data

Going back to the setting of restricted instance classes, we have:

Theorem (A., Bourhis, Senellart, ICALP'15, PODS'16)

For any fixed MSO query $\mathbf{Q}$ *and treelike instance family* $\mathcal{I}$*, given an instance* $\mathbf{I}$ *in* $\mathcal{I}$*, we can compute the provenance of* $\mathbf{Q}$ *on* $\mathbf{I}$ *in PTIME as a d-SDNNF circuit*

Can we get a weaker representation (uOBDD, etc.)? **Open**

What happens on high-treewidth instances?

Theorem (A., Bourhis, Senellart, PODS'16; A., Monet, Senellart, ICDT'18)

For the "two incident facts" query $\mathbf{Q_2}$*, given an instance* $\mathbf{I}$*, any d-SDNNF representation of the provenance of* $\mathbf{Q_2}$ *on* $\mathbf{I}$ *is* ***exponential****: in* $\Omega\left(2^{tw(\mathbf{I})^{1/d}}\right)$ *for some* $\mathbf{d} \geq 1$

For which other queries is this true? Mostly **open** (some results for connected $\mathrm{UCQ}^{\neq}$)

What about more expressive circuit formalisms (d-DNNF)? **Open**

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from knowledge compilation...

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from **knowledge compilation**...

- ... to **SDNNF**, for the combined FPRAS for **bounded-hypertreewidth SJFCQs** (PODS'23)

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from **knowledge compilation**...

- ... to **SDNNF**, for the combined FPRAS for **bounded-hypertreewidth SJFCQs** (PODS'23)
- ... to **nOBDDs** for the combined FPRAS for **one-way paths** on **DAGs** (ICDT'24)

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from **knowledge compilation**…

- … to **SDNNF**, for the combined FPRAS for **bounded-hypertreewidth SJFCQs** (PODS'23)
- … to **nOBDDs** for the combined FPRAS for **one-way paths** on **DAGs** (ICDT'24)
- … to β**-acyclic lineages**, for the exact algorithms for **one-way paths** on **downward trees** and for **connected queries** on **two-way paths** (PODS'17)

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from **knowledge compilation**...

- ... to **SDNNF**, for the combined FPRAS for **bounded-hypertreewidth SJFCQs** (PODS'23)
- ... to **nOBDDs** for the combined FPRAS for **one-way paths** on **DAGs** (ICDT'24)
- ... to **β-acyclic lineages**, for the exact algorithms for **one-way paths** on **downward trees** and for **connected queries** on **two-way paths** (PODS'17)

Many combined complexity **lower bounds** for PQE also apply to **knowledge compilation**:

- **Exponential** lower bounds on **DNNF provenance representations** of **one-way path** queries on **arbitrary** instance graphs (ICDT'24)
  - $\rightarrow$ Same applies to all cases where we show that there is (conditionally) no FPRAS

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from knowledge compilation…

- … to SDNNF, for the combined FPRAS for bounded-hypertreewidth SJFCQs (PODS'23)
- … to nOBDDs for the combined FPRAS for one-way paths on DAGs (ICDT'24)
- … to β-acyclic lineages, for the exact algorithms for one-way paths on downward trees and for connected queries on two-way paths (PODS'17)

Many combined complexity lower bounds for PQE also apply to knowledge compilation:

- Exponential lower bounds on DNNF provenance representations of one-way path queries on arbitrary instance graphs (ICDT'24)
  - $\rightarrow$ Same applies to all cases where we show that there is (conditionally) no FPRAS
- Open if the classes above (SDNNF, nOBDDs, β-acyclic) are "as small as possible"

Combined complexity of provenance computation

Many combined complexity upper bounds for PQE come from **knowledge compilation**...

- ... to **SDNNF**, for the combined FPRAS for **bounded-hypertreewidth SJFCQs** (PODS'23)
- ... to **nOBDDs** for the combined FPRAS for **one-way paths** on **DAGs** (ICDT'24)
- ... to **β-acyclic lineages**, for the exact algorithms for **one-way paths** on **downward trees** and for **connected queries** on **two-way paths** (PODS'17)

Many combined complexity **lower bounds** for PQE also apply to **knowledge compilation**:

- **Exponential** lower bounds on **DNNF provenance representations** of **one-way path** queries on **arbitrary** instance graphs (ICDT'24)
  - $\rightarrow$ Same applies to all cases where we show that there is (conditionally) no FPRAS
- **Open** if the classes above (SDNNF, nOBDDs, β-acyclic) are "as small as possible"

Table of contents

Conclusion

We have seen two frameworks for query explanation via the Boolean provenance, i.e., via the possible worlds that satisfy the query:

- Model counting: unweighted (UR) or weighted (PQE)
- Knowledge compilation: representing the Boolean provenance in tractable circuit classes

Conclusion

We have seen two frameworks for query explanation via the Boolean provenance, i.e., via the possible worlds that satisfy the query:

- Model counting: unweighted (UR) or weighted (PQE)
- Knowledge compilation: representing the Boolean provenance in tractable circuit classes

We have studied this in several settings:

- Data complexity for SJFCQs, UCQs, hom-closed queries, etc.
- Data complexity on restricted instance families (treelike or not)
- Combined complexity

Open problems

General research directions on these topics:

- Connections between these two frameworks (intensional-extensional conjecture? lower bound techniques?)
- Connections to other aggregate queries? (Shapley value, etc.)
- Other provenance uses? semirings, enumeration, incremental maintenance...

Open problems

General research directions on these topics:

- Connections between these two frameworks (intensional-extensional conjecture? lower bound techniques?)
- Connections to other aggregate queries? (Shapley value, etc.)
- Other provenance uses? semirings, enumeration, incremental maintenance...

Many open questions throughout the talk:

- Queries with inequalities, negation, first-order queries: (approximate) PQE?
- Can we show dichotomies on approximation for unbounded queries (RPQs...)
- Understanding higher-arity and uniform reliability where it is still open
- Unbounded queries: what if some relations are non-probabilistic?
- Are there joint criteria for the tractability of instances and queries?

Open problems

General research directions on these topics:

- Connections between these two frameworks (intensional-extensional conjecture? lower bound techniques?)
- Connections to other aggregate queries? (Shapley value, etc.)
- Other provenance uses? semirings, enumeration, incremental maintenance...

Many open questions throughout the talk:

- Queries with inequalities, negation, first-order queries: (approximate) PQE?
- Can we show dichotomies on approximation for unbounded queries (RPQs...)
- Understanding higher-arity and uniform reliability where it is still open
- Unbounded queries: what if some relations are non-probabilistic?
- Are there joint criteria for the tractability of instances and queries?

Thanks for your attention!

Amarilli, A., Bourhis, P., and Senellart, P. (2015).
Provenance circuits for trees and treelike instances.
In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2016).
Tractable lineages on treelike instances: Limits and extensions.
In *PODS*.

Amarilli, A. and Ceylan, I. I. (2020).
A dichotomy for homomorphism-closed queries on probabilistic graphs.
In *ICDT*.

Amarilli, A. and Ceylan, I. I. (2022).
The dichotomy of evaluating homomorphism-closed queries on probabilistic graphs.
LMCS.

Amarilli, A. and Kimelfeld, B. (2021).
Uniform Reliability of Self-Join-Free Conjunctive Queries.
In *ICDT*.

Amarilli, A. and Kimelfeld, B. (2022).
Uniform Reliability of Self-Join-Free Conjunctive Queries.
LMCS.

Amarilli, A., Monet, M., and Senellart, P. (2017).
Conjunctive queries on probabilistic graphs: Combined complexity.
In *PODS*.

Amarilli, A., van Bremen, T., and Meel, K. S. (2024).
Conjunctive queries on probabilistic graphs: The limits of approximability.
In *ICDT*.

Arenas, M., Croquevielle, L. A., Jayaram, R., and Riveros, C. (2021a).
When is approximate counting for conjunctive queries tractable?
In *STOC*.

Arenas, M., Croquevielle, L. A., Jayaram, R., and Riveros, C. (2021b).
#NFA admits an FPRAS: Efficient enumeration, counting, and uniform generation for logspace classes.
JACM, 68(6).

Chekuri, C. and Chuzhoy, J. (2016).
Polynomial bounds for the grid-minor theorem.
JACM, 63(5).

Cohen, S., Kimelfeld, B., and Sagiv, Y. (2009).
Running tree automata on probabilistic XML.
In *PODS*.

Dalvi, N. and Suciu, D. (2004).
Efficient query evaluation on probabilistic databases.
In *VLDB*.

Dalvi, N. and Suciu, D. (2007).
The dichotomy of conjunctive queries on probabilistic structures.
In *PODS*.

Dalvi, N. and Suciu, D. (2012).
The dichotomy of probabilistic inference for unions of conjunctive queries.
J. ACM, 59(6).

Jha, A. and Suciu, D. (2011).
Knowledge compilation meets database theory: Compiling queries to decision diagrams.
In *ICDT*.

Jha, A. and Suciu, D. (2013).
Knowledge compilation meets database theory: Compiling queries to decision diagrams.
TCS, 52(3).

Kenig, B. and Suciu, D. (2021).
A dichotomy for the generalized model counting problem for unions of conjunctive queries.
In *PODS*.

van Bremen, T. and Meel, K. S. (2023).
Probabilistic query evaluation: The combined FPRAS landscape.
In *PODS*.