Knowledge Compilation for Queries on Probabilistic Graphs

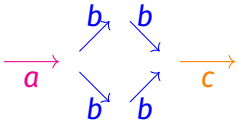
Antoine Amarilli[1]

May 28, 2026

[1]Inria Lille

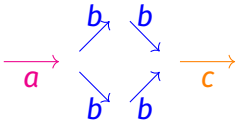
Queries on graphs

- Fix: a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- Graph: a directed graph with edges labeled in Σ



Queries on graphs

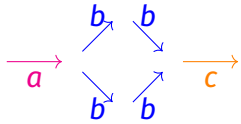
- Fix: a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- Graph: a directed graph with edges labeled in Σ



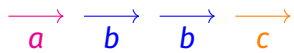
A (Boolean) graph query is simply a function from graphs on Σ to $\{\top, \bot\}$

Queries on graphs

- **Fix:** a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- **Graph:** a directed graph with edges labeled in Σ

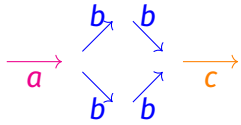


A (Boolean) **graph query** is simply a function from graphs on Σ to $\{\top, \bot\}$

- **Conjunctive query:** existence of explicit pattern, *e.g.*, 

Queries on graphs

- **Fix:** a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- **Graph:** a directed graph with edges labeled in Σ

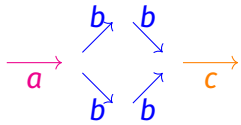


A (Boolean) **graph query** is simply a function from graphs on Σ to $\{\top, \bot\}$

- **Conjunctive query:** existence of explicit pattern, *e.g.*,
- **Union of conjunctive queries:** disjunction of CQs

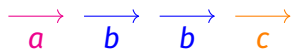
Queries on graphs

- **Fix:** a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- **Graph:** a directed graph with edges labeled in Σ



A (Boolean) **graph query** is simply a function from graphs on Σ to $\{\top, \bot\}$

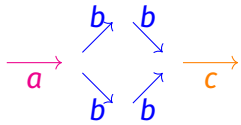
- **Conjunctive query:** existence of explicit pattern, *e.g.*,



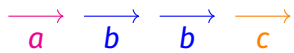
- **Union of conjunctive queries:** disjunction of CQs
- **Regular path query (RPQ):** existence of a walk matching a **regex**, *e.g.*, $a\ b^*\ c$

Queries on graphs

- **Fix:** a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- **Graph:** a directed graph with edges labeled in Σ



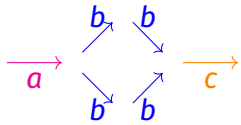
A (Boolean) **graph query** is simply a function from graphs on Σ to $\{\top, \bot\}$

- **Conjunctive query:** existence of explicit pattern, *e.g.,* 

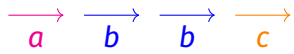
- **Union of conjunctive queries:** disjunction of CQs
- **Regular path query (RPQ):** existence of a walk matching a **regex**, *e.g.,* $a\ b^*\ c$
- **Datalog**, queries closed under homomorphisms, etc.

Queries on graphs

- Fix: a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- Graph: a directed graph with edges labeled in Σ



A (Boolean) graph query is simply a function from graphs on Σ to $\{\top, \bot\}$

- Conjunctive query: existence of explicit pattern, *e.g.,* 

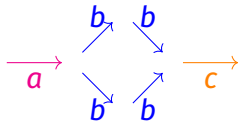
- Union of conjunctive queries: disjunction of CQs
- Regular path query (RPQ): existence of a walk matching a regex, *e.g.,* $a\ b^*\ c$
- Datalog, queries closed under homomorphisms, etc.

Query evaluation problem:

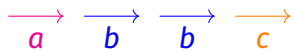
- Input: a query Q from some class, a graph G
- Output: does G satisfy Q?

Queries on graphs

- **Fix:** a signature Σ of edge labels, e.g., $\Sigma = \{a, b, c\}$
- **Graph:** a directed graph with edges labeled in Σ



A (Boolean) **graph query** is simply a function from graphs on Σ to $\{\top, \bot\}$

- **Conjunctive query:** existence of explicit pattern, *e.g.,* 

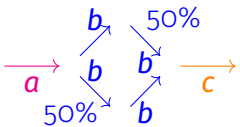
- **Union of conjunctive queries:** disjunction of CQs
- **Regular path query (RPQ):** existence of a walk matching a **regex**, *e.g.,* $a\ b^*\ c$
- **Datalog**, queries closed under homomorphisms, etc.

Query evaluation problem:

- **Input:** a query Q from some class, a graph G
- **Output:** does G satisfy Q?
- **Combined complexity:** as a function of Q and G
- **Data complexity** for fixed Q: as a function of G only

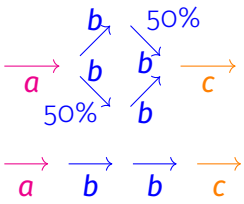
Queries on probabilistic graphs

- Probabilistic graph: edges carry a probability along with their label in Σ
- Each edge is present with the indicated probability, assuming independence
- Concise representation of a probability distribution on the subgraphs



Queries on probabilistic graphs

- **Probabilistic graph:** edges carry a **probability** along with their label in Σ
- Each edge is present with the indicated probability, assuming **independence**
- Concise representation of a probability distribution on the subgraphs

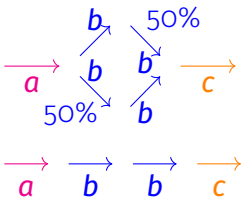


Probabilistic query evaluation problem:

- **Input:** a query Q from some class, a **probabilistic graph** G
- **Output:** the **probability** that G satisfies Q
  - $\rightarrow$ total probability of the subgraphs where Q holds
- Again **combined complexity** and **data complexity**

Queries on probabilistic graphs

- **Probabilistic graph:** edges carry a **probability** along with their label in Σ
- Each edge is present with the indicated probability, assuming **independence**
- Concise representation of a probability distribution on the subgraphs



Probabilistic query evaluation problem:

- **Input:** a query Q from some class, a **probabilistic graph** G
- **Output:** the **probability** that G satisfies Q
  - $\rightarrow$ total probability of the subgraphs where Q holds
- Again **combined complexity** and **data complexity**

We can always **consider all subgraphs** (exponential), but can we do **better**?

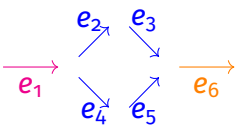
$\rightarrow$ **Exact computation**, or **approximate computation** (FPRAS):

- $\rightarrow$ Randomized algorithm with probability $> 3/4$ that computes a probability p such that the true probability p^* satisfies $(1-\epsilon)p^* \leq p \leq (1+\epsilon)p^*$, in PTIME in the input and in ϵ^{-1}

Provenance for queries on graphs

The (Boolean) provenance of a query Q on a graph G with edge set E is the Boolean function $\mathrm{Prov}(Q, G)$ on variable set E that evaluates to true on E' precisely when $G_{|E'}$ satisfies Q

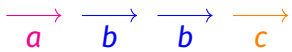
$Q =$ 

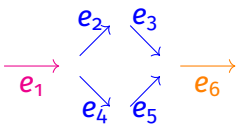
$G =$ 

$\mathrm{Prov}(Q, G) = e_1 \wedge ((e_2 \wedge e_3) \vee (e_4 \wedge e_5)) \wedge e_6$

Provenance for queries on graphs

The (Boolean) provenance of a query Q on a graph G with edge set E is the Boolean function $\mathrm{Prov}(Q, G)$ on variable set E that evaluates to true on E' precisely when $G_{|E'}$ satisfies Q

$Q =$ 

$G =$ 

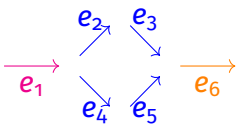
$\mathrm{Prov}(Q, G) = e_1 \wedge ((e_2 \wedge e_3) \vee (e_4 \wedge e_5)) \wedge e_6$

How to do probabilistic query evaluation on graphs via knowledge compilation?

Provenance for queries on graphs

The (Boolean) provenance of a query Q on a graph G with edge set E is the Boolean function $\mathrm{Prov}(Q, G)$ on variable set E that evaluates to true on E' precisely when $G_{|E'}$ satisfies Q

$Q =$ 

$G =$ 

$\mathrm{Prov}(Q, G) = e_1 \wedge ((e_2 \wedge e_3) \vee (e_4 \wedge e_5)) \wedge e_6$

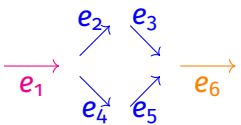
How to do probabilistic query evaluation on graphs via knowledge compilation?

- Compute a representation of the provenance as a tractable circuit
  - → Exact: OBDDs, FBDDs, d-DNNFs, d-Ds
  - → Approximate: nOBDDs, nFBDDs, DNNFs

Provenance for queries on graphs

The (Boolean) provenance of a query Q on a graph G with edge set E is the Boolean function $\mathrm{Prov}(Q, G)$ on variable set E that evaluates to true on E' precisely when $G_{|E'}$ satisfies Q

$Q =$

$G =$ 

$\mathrm{Prov}(Q, G) = e_1 \wedge ((e_2 \wedge e_3) \vee (e_4 \wedge e_5)) \wedge e_6$

How to do probabilistic query evaluation on graphs via knowledge compilation?

- Compute a representation of the provenance as a tractable circuit
  - → Exact: OBDDs, FBDDs, d-DNNFs, d-Ds
  - → Approximate: nOBDDs, nFBDDs, DNNFs
- Use tractable (approximate) model counting on the circuit

Data complexity

CQs and UCQs

For exact probabilistic query evaluation, remember Mikaël's talk:

- Dalvi-Suciu dichotomy between safe UCQs (in FP) and unsafe UCQs (#P-hard)
- Sometimes doable via knowledge compilation
  → (Partial) characterizations: OBDDs, FBDDs, d-DNNFs, d-Ds
- Open whether d-Ds always suffice

CQs and UCQs

For exact probabilistic query evaluation, remember Mikaël's talk:

- Dalvi-Suciu dichotomy between safe UCQs (in FP) and unsafe UCQs (#P-hard)
- Sometimes doable via knowledge compilation
  - → (Partial) characterizations: OBDDs, FBDDs, d-DNNFs, d-Ds
- Open whether d-Ds always suffice

What about approximate probabilistic query evaluation?

CQs and UCQs

For exact probabilistic query evaluation, remember Mikaël's talk:

- Dalvi-Suciu dichotomy between safe UCQs (in FP) and unsafe UCQs (#P-hard)
- Sometimes doable via knowledge compilation
  - → (Partial) characterizations: OBDDs, FBDDs, d-DNNFs, d-Ds
- Open whether d-Ds always suffice

What about approximate probabilistic query evaluation? every UCQ has an FPRAS

CQs and UCQs

For exact probabilistic query evaluation, remember Mikaël's talk:

- Dalvi-Suciu dichotomy between safe UCQs (in FP) and unsafe UCQs (#P-hard)
- Sometimes doable via knowledge compilation
  → (Partial) characterizations: OBDDs, FBDDs, d-DNNFs, d-Ds
- Open whether d-Ds always suffice

What about approximate probabilistic query evaluation? every UCQ has an FPRAS

- Compute all matches of the UCQ Q on the graph G
  → Each match has size $\leq |Q|$ so at most $|G|^{|Q|}$ matches: poly in data complexity
- Write this as a DNF representation of the provenance
  → DNF = disjunctive normal form
- Use the Karp-Luby-Madras algorithm to approximate #SAT on the DNF

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation?

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is...

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is... open!
  → Open whether st-reliability admits an FPRAS

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is... open!
  → Open whether st-reliability admits an FPRAS
- Infinite infix-free RPQs are no easier than st-reliability (some are equivalent)

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is... open!
  - → Open whether st-reliability admits an FPRAS
- Infinite infix-free RPQs are no easier than st-reliability (some are equivalent)
- For some RPQs, there is no FPRAS assuming RP $\neq$ NP *e.g.*, $a\ a\ (b^8\ a)^*\ a$
  - → Reduction from counting vertex covers, aka #MONOTONE-2-CNF

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is... open!
  - → Open whether st-reliability admits an FPRAS
- Infinite infix-free RPQs are no easier than st-reliability (some are equivalent)
- For some RPQs, there is no FPRAS assuming RP $\neq$ NP *e.g.*, $a\ a\ (b^8\ a)^*\ a$
  - → Reduction from counting vertex covers, aka #MONOTONE-2-CNF

Open: are there approximable homomorphism-closed queries beyond UCQs?

Beyond UCQs

What about more expressive queries beyond UCQs? e.g., RPQs, Datalog...

For exact probabilistic query evaluation? always #P-hard

→ All homomorphism-closed queries on graphs are #P-hard except UCQs

For approximate probabilistic query evaluation, for RPQs:

- RPQs with finite languages are UCQs so approximable *e.g.*, $L = a\ b$
- The RPQ $a\ b^*\ c$ is... open!
  - → Open whether st-reliability admits an FPRAS
- Infinite infix-free RPQs are no easier than st-reliability (some are equivalent)
- For some RPQs, there is no FPRAS assuming RP $\neq$ NP *e.g.*, $a\ a\ (b^8\ a)^*\ a$
  - → Reduction from counting vertex covers, aka #MONOTONE-2-CNF

Open: are there approximable homomorphism-closed queries beyond UCQs?

Open: what about lineage lower bounds?

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

- Monadic second-order logic (MSO): very general query language
  → generalizes UCQs, RPQs, guarded Datalog
- Treewidth measures how close an instance is to a tree

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

- Monadic second-order logic (MSO): very general query language
  → generalizes UCQs, RPQs, guarded Datalog
- Treewidth measures how close an instance is to a tree

For any fixed MSO query Q on bounded-treewidth graphs, we can compute a d-SDNNF circuit for the provenance (very similar to Petr's talk)

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

- Monadic second-order logic (MSO): very general query language
  → generalizes UCQs, RPQs, guarded Datalog
- Treewidth measures how close an instance is to a tree

For any fixed MSO query Q on bounded-treewidth graphs, we can compute a d-SDNNF circuit for the provenance (very similar to Petr's talk)

For some queries (open: which ones?), bounding the treewidth is necessary:

- Computational hardness subject to technical assumptions
- d-SDNNF lower bounds; open whether we can show stronger bounds

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

- Monadic second-order logic (MSO): very general query language
  → generalizes UCQs, RPQs, guarded Datalog
- Treewidth measures how close an instance is to a tree

For any fixed MSO query Q on bounded-treewidth graphs, we can compute a d-SDNNF circuit for the provenance (very similar to Petr's talk)

For some queries (open: which ones?), bounding the treewidth is necessary:

- Computational hardness subject to technical assumptions
- d-SDNNF lower bounds; open whether we can show stronger bounds

Open: imposing tractability requirements on both the query and the graph?

Restricting the data

Can we make probabilistic query evaluation tractable by restricting the data?

For exact probabilistic query evaluation, recall what Mikaël didn't have time to do

- Monadic second-order logic (MSO): very general query language
  → generalizes UCQs, RPQs, guarded Datalog
- Treewidth measures how close an instance is to a tree

For any fixed MSO query Q on bounded-treewidth graphs, we can compute a d-SDNNF circuit for the provenance (very similar to Petr's talk)

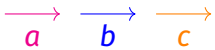
For some queries (open: which ones?), bounding the treewidth is necessary:

- Computational hardness subject to technical assumptions
- d-SDNNF lower bounds; open whether we can show stronger bounds

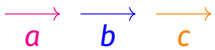
Open: imposing tractability requirements on both the query and the graph?
Open: ensuring the existence of an FPRAS?

Combined complexity

Self-join-free CQs

A CQ is self-join-free if it has no repeated labels, *e.g.*, 

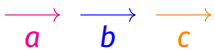
Self-join-free CQs

A CQ is **self-join-free** if it has no repeated labels, *e.g.*, 

For **exact** probabilistic query evaluation, recall what Mikaël presented:

- For **non-hierarchical** self-join-free CQs, the task is **#P-hard** (already in data)
- For **hierarchical** self-join-free CQs, tractable **also in combined complexity**
  → We can compute an **OBDD** for the provenance (or a d-D, cf Mikaël's proof)

Self-join-free CQs

A CQ is self-join-free if it has no repeated labels, *e.g.*, 

For exact probabilistic query evaluation, recall what Mikaël presented:

- For non-hierarchical self-join-free CQs, the task is #P-hard (already in data)
- For hierarchical self-join-free CQs, tractable also in combined complexity
  → We can compute an OBDD for the provenance (or a d-D, cf Mikaël's proof)

For approximate probabilistic query evaluation, result by Kuldeep and Tim:
There is a combined FPRAS for self-join-free CQs of bounded hypertreewidth

→ Amounts to computing a SDNNF of the query provenance

CQs with self-joins and UCQs

For exact probabilistic query evaluation:

- For unsafe UCQs, the problem is #P-hard (already in data)
- For safe UCQs, the problem is...

CQs with self-joins and UCQs

For exact probabilistic query evaluation:

- For unsafe UCQs, the problem is #P-hard (already in data)
- For safe UCQs, the problem is... open
  → Relates to the complexity in the query of the Dalvi-Suciu algorithm

CQs with self-joins and UCQs

For exact probabilistic query evaluation:

- For unsafe UCQs, the problem is #P-hard (already in data)
- For safe UCQs, the problem is... open
  → Relates to the complexity in the query of the Dalvi-Suciu algorithm

For approximate probabilistic query evaluation:

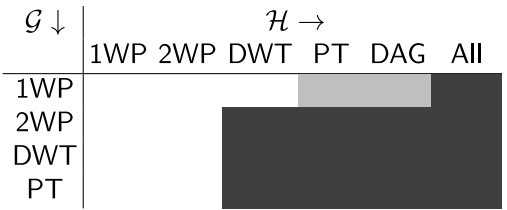
- One-way path CQs with self-joins have no FPRAS assuming RP $\neq$ NP
- Open: what about one-way path CQs on a unary alphabet ($\Sigma = \{a\}$)

Restricting the data

Can we make combined probabilistic query evaluation more tractable?

Restricting the data

Can we make combined probabilistic query evaluation more tractable?



(A) Complexity of $\mathsf{PHom}_{\mathsf{L}}(\mathcal{G},\mathcal{H})$.

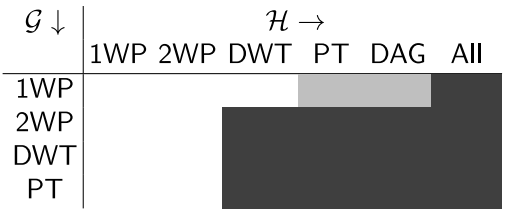
(B) Complexity of $\mathsf{PHom}_{\not\mathsf{L}}(\mathcal{G},\mathcal{H})$.

1WP = one-way path; 2WP = two-way path; DWT = downward tree; PT = polytree

white = exact is in FP; gray = #P-hard but FPRAS; black = conditionally no FPRAS

Restricting the data

Can we make combined probabilistic query evaluation more tractable?



(A) Complexity of $\mathsf{PHom}_{\mathsf{L}}(\mathcal{G},\mathcal{H})$.

(B) Complexity of $\mathsf{PHom}_{\not\mathsf{L}}(\mathcal{G},\mathcal{H})$.

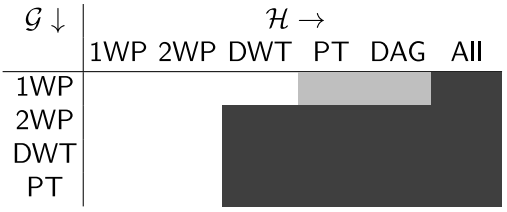
1WP = one-way path; 2WP = two-way path; DWT = downward tree; PT = polytree

white = exact is in FP; gray = #P-hard but FPRAS; black = conditionally no FPRAS

- Tractable exact cases use d-SDNNFs and β-acyclic lineages ($\rightarrow$ dec-DNNF)
- Interesting tractable approximate case: one-way paths on acyclic graphs
  - $\rightarrow$ Amounts to computing an nOBDD

Restricting the data

Can we make combined probabilistic query evaluation more tractable?



(A) Complexity of $\mathsf{PHom}_{\mathsf{L}}(\mathcal{G}, \mathcal{H})$.

(B) Complexity of $\mathsf{PHom}_{\not\mathsf{L}}(\mathcal{G}, \mathcal{H})$.

1WP = one-way path; 2WP = two-way path; DWT = downward tree; PT = polytree

white = exact is in FP; gray = #P-hard but FPRAS; black = conditionally no FPRAS

- Tractable exact cases use d-SDNNFs and β-acyclic lineages ($\rightarrow$ dec-DNNF)
- Interesting tractable approximate case: one-way paths on acyclic graphs
  - $\rightarrow$ Amounts to computing an nOBDD

Inapproximable cases come with DNNF lower bounds

Conclusion

Conclusion and open problems

What remains to completely classify the complexity landscape?

Conclusion and open problems

What remains to completely classify the complexity landscape?

- Lineage and approximability lower bounds for recursive queries
  → **Open:** does st-reliability admit DNNFs? an FPRAS?

Conclusion and open problems

What remains to completely classify the complexity landscape?

- Lineage and approximability lower bounds for recursive queries
  → **Open:** does st-reliability admit DNNFs? an FPRAS?
- Knowledge compilation proof (via d-Ds?) of the Dalvi-Suciu dichotomy

Conclusion and open problems

What remains to completely classify the complexity landscape?

- Lineage and approximability lower bounds for recursive queries
  → **Open:** does st-reliability admit DNNFs? an FPRAS?
- Knowledge compilation proof (via d-Ds?) of the Dalvi-Suciu dichotomy
- Imposing finer tractability requirements on the query and the graph

Conclusion and open problems

What remains to completely classify the complexity landscape?

- Lineage and approximability lower bounds for recursive queries
  - → **Open:** does st-reliability admit DNNFs? an FPRAS?
- Knowledge compilation proof (via d-Ds?) of the Dalvi-Suciu dichotomy
- Imposing finer tractability requirements on the query and the graph

Other questions:

- Generalizing from graphs to databases? (not everything generalizes)
- Links to other counting problems? Shapley value, etc.?
  - Resilience problem: relates to the "most likely countermodel" question

Conclusion and open problems

What remains to completely classify the complexity landscape?

- Lineage and approximability lower bounds for recursive queries
  - → **Open:** does st-reliability admit DNNFs? an FPRAS?
- Knowledge compilation proof (via d-Ds?) of the Dalvi-Suciu dichotomy
- Imposing finer tractability requirements on the query and the graph

Other questions:

- Generalizing from graphs to databases? (not everything generalizes)
- Links to other counting problems? Shapley value, etc.?
  - Resilience problem: relates to the "most likely countermodel" question

Thanks for your attention!

References i


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.

References ii

Amarilli, A. and Ceylan, I. I. (2022).
The dichotomy of evaluating homomorphism-closed queries on probabilistic graphs.
LMCS, 18.

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

Amarilli, A., Monet, M., and Senellart, P. (2018).
Connecting width and structure in knowledge compilation.
In ICDT.

References iii

- Amarilli, A., Monet, M., and Suciu, D. (2024).
  The non-cancelling intersections conjecture.
  Preprint: https://arxiv.org/abs/2401.16210.
- Amarilli, A., van Bremen, T., Gaspard, O., and Meel, K. S. (2025).
  Approximating queries on probabilistic graphs.
  LMCS.
- Arenas, M., Croquevielle, L. A., Jayaram, R., and Riveros, C. (2021).
  When is approximate counting for conjunctive queries tractable?
  In STOC.

References iv

- Dalvi, N. and Suciu, D. (2007).
  Efficient query evaluation on probabilistic databases.
  VLDBJ, 16(4).
- Dalvi, N. and Suciu, D. (2013).
  The dichotomy of probabilistic inference for unions of conjunctive queries.
  JACM, 59(6).
- Jha, A. and Suciu, D. (2013).
  Knowledge compilation meets database theory: Compiling queries to decision diagrams.
  Theory of Computing Systems, 52(3).

References v

- Meel, K. S. and de Colnet, A. (2026).
  #cfg and #dnnf admit fpras.
  In Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 5978–6010. SIAM.
- Suciu, D., Olteanu, D., Ré, C., and Koch, C. (2011).
  Probabilistic databases.
  Synthesis lectures on data management, 3(2).
  ISBN: 978-1608456802.
- van Bremen, T. and Meel, K. S. (2023).
  Probabilistic query evaluation: The combined FPRAS landscape.
  In PODS.