A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs

Antoine Amarilli\textsuperscript{1} and İsmail İlkan Ceylan\textsuperscript{2}
March 30, 2020

\textsuperscript{1}Télécom Paris
\textsuperscript{2}University of Oxford
In this talk, we manage data represented as a labeled graph.
Uncertain data management

In this talk, we manage data represented as a labeled graph

<table>
<thead>
<tr>
<th>WorksAt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antoine</td>
</tr>
<tr>
<td>Antoine</td>
</tr>
<tr>
<td>Benny</td>
</tr>
<tr>
<td>Benny</td>
</tr>
<tr>
<td>İsmail</td>
</tr>
</tbody>
</table>
Uncertain data management

In this talk, we manage data represented as a labeled graph

<table>
<thead>
<tr>
<th>WorksAt</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Antoine</td>
<td>Télécom Paris</td>
</tr>
<tr>
<td>Antoine</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Technion</td>
</tr>
<tr>
<td>İsmail</td>
<td>U. Oxford</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MemberOf</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Télécom Paris</td>
<td>ParisTech</td>
</tr>
<tr>
<td>Télécom Paris</td>
<td>IP Paris</td>
</tr>
<tr>
<td>Paris Sud</td>
<td>IP Paris</td>
</tr>
<tr>
<td>Paris Sud</td>
<td>Paris-Saclay</td>
</tr>
<tr>
<td>Technion</td>
<td>CESAER</td>
</tr>
</tbody>
</table>
Uncertain data management

In this talk, we manage data represented as a labeled graph.

### WorksAt

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antoine</td>
<td>Télécom Paris</td>
</tr>
<tr>
<td>Antoine</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Technion</td>
</tr>
<tr>
<td>İsmail</td>
<td>U. Oxford</td>
</tr>
</tbody>
</table>

### MemberOf

<table>
<thead>
<tr>
<th>Institution</th>
<th>ParisTech</th>
<th>IP Paris</th>
<th>Paris-Saclay</th>
<th>CESAER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Télécom Paris</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paris Sud</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paris Sud</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>İsmail</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U. Oxford</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Uncertain data management

In this talk, we manage data represented as a labeled graph.

**WorksAt**

- Antoine: Télécom Paris
- Antoine: Paris Sud
- Benny: Paris Sud
- Benny: Technion
- İsmail: U. Oxford

**MemberOf**

- Télécom Paris: ParisTech
- Télécom Paris: IP Paris
- Paris Sud: IP Paris
- Paris Sud: Paris-Saclay
- Technion: CESAER

A. → Télécom Paris → ParisTech
B. → Paris Sud → IP Paris
İ. → U. Oxford → CESAER
In this talk, we manage data represented as a labeled graph.
Uncertain data management

In this talk, we manage data represented as a labeled graph.

WorksAt

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Antoine</td>
<td>Télécom Paris</td>
</tr>
<tr>
<td>Antoine</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Paris Sud</td>
</tr>
<tr>
<td>Benny</td>
<td>Technion</td>
</tr>
<tr>
<td>İsmail</td>
<td>U. Oxford</td>
</tr>
</tbody>
</table>

MemberOf

<table>
<thead>
<tr>
<th>Télécom Paris</th>
<th>ParisTech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Télécom Paris</td>
<td>IP Paris</td>
</tr>
<tr>
<td>Paris Sud</td>
<td>IP Paris</td>
</tr>
<tr>
<td>Paris Sud</td>
<td>Paris-Saclay</td>
</tr>
<tr>
<td>Technion</td>
<td>CESAER</td>
</tr>
<tr>
<td>İ.</td>
<td>U. Oxford</td>
</tr>
<tr>
<td></td>
<td>CESAER</td>
</tr>
</tbody>
</table>

→ **Problem:** we are not certain about the true state of the data
Uncertain data model

- Uncertain data model: **TID**, for tuple-independent database
- Each fact (edge) carries a **probability**

A. Télécom Paris → ParisTech

B. Paris Sud → IP Paris

B. Technion → Paris-Saclay

i. U. Oxford → CESAER
Uncertain data model:

- **TID**, for tuple-independent database
- Each fact (edge) carries a probability
- All facts are independent
- Possible world $W$:
  - What is the probability of this possible world?

$$\Pr(W) = \prod_{F \in W} \Pr(F) \times \prod_{F \notin W} \left(1 - \Pr(F)\right)$$
Uncertain data model

A. 80% → Télécom Paris → ParisTech
    10%

B. Paris Sud → IP Paris
    40%
    80%

Technion → Paris-Saclay
    90%
    50%
    90%

U. Oxford → CESAER
    100%

- Uncertain data model: TID, for tuple-independent database
- Each fact (edge) carries a probability
- Each fact exists with its given probability
- All facts are independent
Uncertain data model

• Uncertain data model: TID, for tuple-independent database
• Each fact (edge) carries a probability
• Each fact exists with its given probability
• All facts are independent
• Possible world $W$: subset of facts

A. 80% → Télécom Paris → ParisTech
   10% →

   90% → Paris Sud → IP Paris
   50% →

   40% → Technion

B. 80% →

   90% → Paris-Saclay
   100% →

i. 100% → U. Oxford → CESAER
Uncertain data model:

- Uncertain data model: TID, for tuple-independent database
- Each fact (edge) carries a probability
- Each fact exists with its given probability
- All facts are independent
- Possible world $W$: subset of facts
Uncertain data model:

- Uncertain data model: **TID**, for tuple-independent database
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world** \( W \): subset of facts
- What is the **probability** of this possible world?

A. → Télécom Paris \( \rightarrow \) ParisTech

B. → Technion
    - → Paris-Saclay
    - → IP Paris

i. → U. Oxford

j. → CESAER
Uncertain data model

- Uncertain data model: **TID**, for tuple-independent database
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world** $W$: subset of facts
- What is the **probability** of this possible world?
Uncertain data model

- Uncertain data model: **TID**, for tuple-independent database
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- Possible world $W$: subset of facts
- What is the **probability** of this possible world? **0.03%**
Uncertain data model

- Uncertain data model: TID, for tuple-independent database
- Each fact (edge) carries a probability
- Each fact exists with its given probability
- All facts are independent
- Possible world $W$: subset of facts
- What is the probability of this possible world? $0.03\%$

$$\Pr(W) = \left( \prod_{F \in W} \Pr(F) \right) \times \left( \prod_{F \notin W} (1 - \Pr(F)) \right)$$
• **Query**: maps a graph *(without probabilities)* to YES/NO
Queries

- **Query**: maps a graph (**without probabilities**) to YES/NO
- **Conjunctive query (CQ)**: can I find a match of a pattern? e.g., \( x \rightarrow y \rightarrow z \)
Queries

- **Query**: maps a graph (without probabilities) to YES/NO
- **Conjunctive query** (CQ): can I find a match of a pattern? e.g., $x \rightarrow y \rightarrow z$
  - We want a homomorphism from the pattern to the graph (not necessarily injective)

Intuition about homomorphism-closed queries:

- Generalize CQs and UCQs, but also regular path queries (RPQs), Datalog, etc.
- Do not allow for inequalities or negation
- A homomorphism-closed query can be seen as an infinite union of CQs
  - The query is bounded if the union is finite (it is a UCQ), unbounded otherwise
- Allows pretty wild things, e.g., "There is a path whose length is prime"
• **Query:** maps a graph (without probabilities) to YES/NO

• **Conjunctive query (CQ):** can I find a match of a pattern? e.g., \( x \rightarrow y \rightarrow z \)
  → We want a homomorphism from the pattern to the graph (not necessarily injective)

• **Union of conjunctive queries (UCQ):** can I find a match of some pattern?

Intuition about homomorphism-closed queries:
• Generalize CQs and UCQs, but also regular path queries (RPQs), Datalog, etc.
  • Do not allow for inequalities or negation
  • A homomorphism-closed query can be seen as an infinite union of CQs:
    → The query is bounded if the union is finite (it is a UCQ), unbounded otherwise
  • Allows pretty wild things, e.g., "There is a path whose length is prime"
• Query: maps a graph (without probabilities) to YES/NO

• Conjunctive query (CQ): can I find a match of a pattern? e.g., \( x \rightarrow y \rightarrow z \)
  → We want a homomorphism from the pattern to the graph (not necessarily injective)

• Union of conjunctive queries (UCQ): can I find a match of some pattern?
  → Homomorphism-closed query \( Q \): if \( G \) satisfies \( Q \) and \( G \) has a homomorphism to \( G' \)
    then \( G' \) also satisfies \( Q \)
Queries

- **Query**: maps a graph (*without probabilities*) to YES/NO
- **Conjunctive query (CQ)**: can I find a match of a *pattern*? e.g., $x \rightarrow y \rightarrow z$
  → We want a *homomorphism* from the pattern to the graph (not necessarily *injective*)
- **Union of conjunctive queries (UCQ)**: can I find a match of *some pattern*?
  → **Homomorphism-closed query $Q$**: if $G$ satisfies $Q$ and $G$ has a homomorphism to $G'$
    then $G'$ also satisfies $Q$

Intuition about homomorphism-closed queries:
Queries

- **Query**: maps a graph (without probabilities) to YES/NO
- **Conjunctive query (CQ)**: can I find a match of a pattern? e.g., $x \rightarrow y \rightarrow z$
  → We want a homomorphism from the pattern to the graph (not necessarily injective)
- **Union of conjunctive queries (UCQ)**: can I find a match of some pattern?
  → Homomorphism-closed query $Q$: if $G$ satisfies $Q$ and $G$ has a homomorphism to $G'$
  then $G'$ also satisfies $Q$

Intuition about homomorphism-closed queries:

- Generalize **CQs** and **UCQs**, but also regular path queries (RPQs), Datalog, etc.
Queries

- **Query**: maps a graph (without probabilities) to YES/NO
- **Conjunctive query (CQ)**: can I find a match of a pattern? e.g., $x \rightarrow y \rightarrow z$ → We want a homomorphism from the pattern to the graph (not necessarily injective)
- **Union of conjunctive queries (UCQ)**: can I find a match of some pattern?
  → Homomorphism-closed query $Q$: if $G$ satisfies $Q$ and $G$ has a homomorphism to $G'$ then $G'$ also satisfies $Q$

Intuition about homomorphism-closed queries:

- Generalize CQs and UCQs, but also regular path queries (RPQs), Datalog, etc.
- Do not allow for inequalities or negation
Queries

- **Query:** maps a graph (without probabilities) to YES/NO

- **Conjunctive query** (CQ): can I find a match of a pattern? e.g., \( x \rightarrow y \rightarrow z \)
  
  → We want a **homomorphism** from the pattern to the graph (not necessarily injective)

- **Union of conjunctive queries** (UCQ): can I find a match of some pattern?
  
  → **Homomorphism-closed query** \( Q \): if \( G \) satisfies \( Q \) and \( G \) has a homomorphism to \( G' \) then \( G' \) also satisfies \( Q \)

Intuition about homomorphism-closed queries:

- Generalize **CQs** and **UCQs**, but also **regular path queries** (RPQs), **Datalog**, etc.

- Do not allow for **inequalities** or **negation**

- A homomorphism-closed query can be seen as an **infinite union of CQs**:
  
  → The query is **bounded** if the union is finite (it is a UCQ), **unbounded** otherwise
Queries

- **Query**: maps a graph (without probabilities) to YES/NO
- **Conjunctive query (CQ)**: can I find a match of a pattern? e.g., $x \rightarrow y \rightarrow z$
  → We want a homomorphism from the pattern to the graph (not necessarily injective)
- **Union of conjunctive queries (UCQ)**: can I find a match of some pattern?
  → Homomorphism-closed query $Q$: if $G$ satisfies $Q$ and $G$ has a homomorphism to $G'$
  then $G'$ also satisfies $Q$

Intuition about homomorphism-closed queries:

- Generalize **CQs** and **UCQs**, but also regular path queries (RPQs), Datalog, etc.
- Do not allow for inequalities or negation
- A homomorphism-closed query can be seen as an infinite union of CQs:
  → The query is **bounded** if the union is finite (it is a UCQ), **unbounded** otherwise
- Allows pretty wild things, e.g., “There is a path whose length is prime”
Problem statement: Probabilistic query evaluation (PQE)

- We fix a query $Q$, for instance the CQ: $x \rightarrow y \rightarrow z$
Problem statement: Probabilistic query evaluation (PQE)

- We **fix** a query $Q$, for instance the CQ: $x \rightarrow y \rightarrow z$
- The **input** is a TID $D$:

```
A. 80% → Télécom Paris → 90% → ParisTech
10%   90%

Paris Sud → 50% → IP Paris
40%  90%

B. 80% → Technion → Paris-Saclay

i. 100% → U. Oxford → CESAER
```
Problem statement: Probabilistic query evaluation (PQE)

• We **fix** a query $Q$, for instance the CQ: $x \rightarrow y \rightarrow z$

• The **input** is a TID $D$:

  A. 80% $\rightarrow$ Télécom Paris → 90% ParisTech
      10% $\rightarrow$ Paris Sud → 50% IP Paris

  B. 80% $\rightarrow$ Technion $\rightarrow$ Paris-Saclay

  i. 100% $\rightarrow$ U. Oxford $\rightarrow$ CESAER

• The **output** is the **total probability** of the worlds which satisfy the query:
Problem statement: Probabilistic query evaluation (PQE)

- We **fix** a query $Q$, for instance the CQ: $x \rightarrow y \rightarrow z$

- The **input** is a TID $D$:

- The **output** is the **total probability** of the worlds which satisfy the query:
  - Formally: $\sum_{W \subseteq D, W \models Q} \Pr(W)$
  - **Intuition**: the probability that the query is true
Problem statement: Probabilistic query evaluation (PQE)

• We fix a query $Q$, for instance the CQ: $x \rightarrow y \rightarrow z$

• The input is a TID $D$:

• The output is the total probability of the worlds which satisfy the query:
  • Formally: $\sum_{W \subseteq D, W \models Q} \Pr(W)$
  → Intuition: the probability that the query is true

→ What is the complexity of the problem $\text{PQE}(Q)$, depending on the query $Q$?
Existing results on PQE

Dichotomy on the **unions of conjunctive queries** (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are **safe** and $\text{PQE}(Q)$ is in **PTIME**
- All others are **unsafe** and $\text{PQE}(Q)$ is **#P-hard**
Existing results on PQE

Dichotomy on the unions of conjunctive queries (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are *safe* and $\text{PQE}(Q)$ is in **PTIME**
- All others are *unsafe* and $\text{PQE}(Q)$ is **#P-hard**

- The CQ $x \rightarrow y \rightarrow z$ is *safe*, but the CQ $x \rightarrow y \rightarrow z \rightarrow w$ is *unsafe*
Existing results on PQE

Dichotomy on the unions of conjunctive queries (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are **safe** and $\text{PQE}(Q)$ is in $\mathcal{PTIME}$
- All others are **unsafe** and $\text{PQE}(Q)$ is $\#P$-hard

- The CQ $x \rightarrow y \rightarrow z$ is **safe**, but the CQ $x \rightarrow y \rightarrow z \rightarrow w$ is **unsafe**

What about **more expressive queries**?
Existing results on PQE

Dichotomy on the **unions of conjunctive queries** (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are **safe** and PQE($Q$) is in **PTIME**
- All others are **unsafe** and PQE($Q$) is **#P-hard**

- The CQ $x \rightarrow y \rightarrow z$ is **safe**, but the CQ $x \rightarrow y \rightarrow z \rightarrow w$ is **unsafe**

What about more expressive queries?

- Work by [Fink and Olteanu, 2016] about **negation**
Existing results on PQE

Dichotomy on the unions of conjunctive queries (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are **safe** and $\text{PQE}(Q)$ is in $\text{PTIME}$
- All others are **unsafe** and $\text{PQE}(Q)$ is $\#P$-**hard**

- The CQ $x \rightarrow y \rightarrow z$ is **safe**, but the CQ $x \rightarrow y \rightarrow z \rightarrow w$ is **unsafe**

What about **more expressive queries**?

- Work by [Fink and Olteanu, 2016] about **negation**
- No work about **recursive queries** (but no works about RPQs, Datalog, etc.)
Existing results on PQE

Dichotomy on the unions of conjunctive queries (UCQs):

**Theorem [Dalvi and Suciu, 2012]**

- Some UCQs $Q$ are **safe** and $\text{PQE}(Q)$ is in $\text{PTIME}$
- All others are **unsafe** and $\text{PQE}(Q)$ is $\#P$-hard

- The CQ $x \rightarrow y \rightarrow z$ is **safe**, but the CQ $x \rightarrow y \rightarrow z \rightarrow w$ is **unsafe**

What about more expressive queries?

- Work by [Fink and Olteanu, 2016] about **negation**
- No work about **recursive queries** (but no works about RPQs, Datalog, etc.)
- Only exception: work on **ontology-mediated query answering** [Jung and Lutz, 2012]
Our result

We study PQE for homomorphism-closed queries and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a safe UCQ (hence bounded) and $\text{PQE}(Q)$ is in PTIME
- In all other cases, $\text{PQE}(Q)$ is $\#P$-hard

• Example: the RPQ $Q$:

  $$
  \exists x (y \land \neg z)
  $$

  It is not equivalent to a UCQ: infinite disjunction $$(i \in \mathbb{N})$$

  Hence, $\text{PQE}(Q)$ is $\#P$-hard

• We do not study the complexity of deciding which case applies

  Depends on how queries are represented
Our result

We study PQE for homomorphism-closed queries and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a safe UCQ (hence bounded) and $\text{PQE}(Q)$ is in $\text{PTIME}$
- In all other cases, $\text{PQE}(Q)$ is $\#P$-hard

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
Our result

We study PQE for homomorphism-closed queries and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a safe UCQ (hence bounded) and $\text{PQE}(Q)$ is in \text{PTIME}
- In all other cases, $\text{PQE}(Q)$ is \#P-hard

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the RPQ $Q$: \[
\begin{array}{c}
\rightarrow \\
(\rightarrow)^* \\
\rightarrow
\end{array}
\]
Our result

We study PQE for homomorphism-closed queries and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a safe UCQ (hence bounded) and $\text{PQE}(Q)$ is in $\text{PTIME}$
- In all other cases, $\text{PQE}(Q)$ is $\#P$-hard

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the RPQ $Q$: $\xymatrix{ \rightarrow & \rightarrow \ar[r] & (\rightarrow \rightarrow)^* }$
  - It is not equivalent to a UCQ: infinite disjunction $\xymatrix{ \rightarrow & \rightarrow \ar[r] & (\rightarrow \rightarrow)^i }$ for all $i \in \mathbb{N}$
Our result

We study PQE for homomorphism-closed queries and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a **safe UCQ** (hence **bounded**) and $\text{PQE}(Q)$ is in **PTIME**
- In all other cases, $\text{PQE}(Q)$ is $#\text{P-hard}$

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the RPQ $Q$: $\rightarrow (\rightarrow)^*$
  - It is **not equivalent to a UCQ**: infinite disjunction $\rightarrow (\rightarrow)^i$ for all $i \in \mathbb{N}$
  - Hence, $\text{PQE}(Q)$ is $#\text{P-hard}$
Our result

We study PQE for **homomorphism-closed queries** and show:

**Theorem**

For any query $Q$ closed under homomorphisms:

- Either $Q$ is equivalent to a **safe UCQ** (hence **bounded**) and $\text{PQE}(Q)$ is in **PTIME**
- In all other cases, $\text{PQE}(Q)$ is $\#P$-**hard**

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the RPQ $Q$: \[ \rightarrow (\rightarrow)^* \]
  - It is **not equivalent to a UCQ**: infinite disjunction \[ \rightarrow (\rightarrow)^i \] for all $i \in \mathbb{N}$
  - Hence, $\text{PQE}(Q)$ is $\#P$-**hard**
- We do not study the **complexity of deciding which case applies**
  - Depends on how queries are **represented**
Proof structure
Basic idea: finding a tight pattern

The challenging part is to show:

**Theorem**

For any query $Q$ closed under homomorphisms and **unbounded**, $\text{PQE}(Q)$ is **#P-hard**
Basic idea: finding a tight pattern

The challenging part is to show:

**Theorem**

For any query $Q$ closed under homomorphisms and **unbounded**, $\text{PQE}(Q)$ is $\#P$-hard

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges $\rightarrow \rightarrow \rightarrow$ such that:

- satisfies $Q$
- violates $Q$
Basic idea: finding a tight pattern

The challenging part is to show:

**Theorem**

For any query \( Q \) closed under homomorphisms and *unbounded*, \( \text{PQE}(Q) \) is \#P-hard

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges \( \rightarrow \rightarrow \rightarrow \) such that:

1. satisfies \( Q \)
2. violates \( Q \)
Basic idea: finding a tight pattern

The challenging part is to show:

**Theorem**

*For any query $Q$ closed under homomorphisms and unbounded, $\text{PQE}(Q)$ is $\#P$-hard*

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges $\rightarrow \rightarrow \rightarrow$ such that:

- satisfies $Q$
- violates $Q$
Basic idea: finding a tight pattern

The challenging part is to show:

**Theorem**

For any query $Q$ closed under homomorphisms and *unbounded*, $PQE(Q)$ is \#P-hard

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges such that:

- satisfies $Q$
- violates $Q$

**Theorem**

Any *unbounded* query closed under homomorphisms has a tight pattern
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the **tight pattern**:

  - satisfies $Q$
  - violates $Q$

  ![Diagram](image.png)

Idea: possible worlds at the left have a path that matches $Q$ iff the corresponding possible world of the TID at the right satisfies $Q$... except we need more from the tight pattern!
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the **tight pattern**:
  - satisfies $Q$
  - violates $Q$
  - but

- We reduce from PQE for the **unsafe** CQ: $Q_o : x \rightarrow y \rightarrow z \rightarrow w$
Using tight patterns to show hardness of PQE

• Fix the query $Q$ and the tight pattern:

$$\begin{array}{c}
\text{satisfies } Q \\
\text{but} \\
\text{violates } Q
\end{array}$$

• We reduce from PQE for the unsafe CQ: $Q_0 : x \to y \to z \to w$

is coded as
Using tight patterns to show hardness of PQE

• Fix the query $Q$ and the **tight pattern**:

- satisfies $Q$
- violates $Q$

• We reduce from PQE for the **unsafe** CQ: $Q_o : x \rightarrow y \rightarrow z \rightarrow w$

- is coded as

**Idea:** possible worlds at the **left** have a path that matches $Q_o$
iff the corresponding possible world of the TID at the **right** satisfies the query $Q$...
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the **tight pattern**:

  - satisfies $Q$
  - violates $Q$

- We reduce from PQE for the **unsafe** CQ: $Q_o : \ x \rightarrow y \rightarrow z \rightarrow w$

Idea: possible worlds at the **left** have a path that matches $Q_o$ iff the corresponding possible world of the TID at the **right** satisfies the query $Q$...
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the tight pattern:

  satisfies $Q$ but violates $Q$

- We reduce from PQE for the unsafe CQ: $Q_o : x \rightarrow y \rightarrow z \rightarrow w$

  is coded as

**Idea:** possible worlds at the left have a path that matches $Q_o$ iff the corresponding possible world of the TID at the right satisfies the query $Q$...
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the tight pattern:
  - satisfies $Q$
  - violates $Q$

- We reduce from PQE for the unsafe CQ: $Q_o : x \rightarrow y \rightarrow z \rightarrow w$

Idea: possible worlds at the left have a path that matches $Q_o$ iff the corresponding possible world of the TID at the right satisfies the query $Q$... except we need more from the tight pattern!
Using tight patterns to show hardness of PQE

- Fix the query $Q$ and the **tight pattern**:
  
  - satisfies $Q$
  
  - violates $Q$

- We reduce from PQE for the **unsafe** CQ: $Q_o : x \rightarrow y \rightarrow z \rightarrow w$

  - $a_1' \rightarrow a_1 \rightarrow b_1 \rightarrow b_1'$
  
  - $a_2' \rightarrow a_2$
  
  - $a_3' \rightarrow a_3 \rightarrow b_2 \rightarrow b_2'$

  - is coded as

**Idea:** possible worlds at the **left** have a path that matches $Q_o$ iff the corresponding possible world of the TID at the **right** satisfies the query $Q$...

... except we need **more** from the tight pattern!
We know that we have a **tight pattern**:

- satisfies $Q$
- violates $Q$

**Case /one.osf:** some iterate violates the query:

- satisfies $Q$
- violates $(i)$

→ Reduce from $PQE(Q/zero.osf)$ as we explained

**Case /two.osf:** all iterates satisfy the query:

- satisfies $Q$
- violates $(i)$

→ Call this an iterable pattern $/one.osf/zero.osf//one.osf/two.osf$
Saving the proof

We know that we have a **tight pattern**:

- Satisfies $Q$
- Violates $Q$

Consider its **iterates**

\[ \text{Case /one.osf:} \]
- Some iterate violates the query:
  - Satisfies $Q$
  - Violates $Q$
  \[ \rightarrow \] Reduce from $\text{PQE}(Q/\text{zero.osf})$

\[ \text{Case /two.osf:} \]
- All iterates satisfy the query:
  - Satisfies $Q$ for all $n \in \mathbb{N}$
  \[ \rightarrow \] Call this an iterable pattern $/\text{one.osf}/\text{zero.osf}/\text{two.osf}$
We know that we have a **tight pattern:**

- satisfies $Q$
- violates $Q$

Consider its **iterates** for each $n \in \mathbb{N}$:

$$\left( \begin{array}{c} \bullet \\ \rightarrow \\ \bullet \\ \leftarrow \\ \bullet \end{array} \right)^n$$
We know that we have a **tight pattern**:

- satisfies $Q$
- violates $Q$

Consider its **iterates** for each $n \in \mathbb{N}$:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^n \rightarrow \bullet$
- but

→ Reduce from $PQE(\mathit{Q}/\mathit{zero.osf})$ as we explained

→ Call this an iterable pattern $/\mathit{one.osf}/\mathit{zero.osf}/\mathit{one.osf}/\mathit{two.osf}$
Saving the proof

We know that we have a **tight pattern**:

- satisfies $Q$
- violates $Q$

Consider its **iterates** for each $n \in \mathbb{N}$:

**Case 1**: some iterate **violates** the query:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^i \rightarrow \bullet$ satisfies $Q$
- but $(\bullet \rightarrow \bullet \leftarrow \bullet)^{i+1} \rightarrow \bullet$ violates $Q$

**Case 2**: all iterates satisfy the query:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^n \rightarrow \bullet$ satisfies $Q$
- for all $n \in \mathbb{N}$
Saving the proof

We know that we have a tight pattern:

- satisfies $Q$
- violates $Q$

but

Consider its iterates for each $n \in \mathbb{N}$:

Case 1: some iterate violates the query:

- $(\rightarrow \leftarrow \rightarrow)^i$ satisfies $Q$
- $(\rightarrow \leftarrow \rightarrow)^{i+1}$ violates $Q$

$\rightarrow$ Reduce from $\text{PQE}(Q_o)$ as we explained

Case 2: all iterates satisfy the query:

- $(\rightarrow \leftarrow \rightarrow)^n$ satisfies $Q$
- $(\rightarrow \leftarrow \rightarrow)^{n+1}$ violates $Q$

$\rightarrow$ Call this an iterable pattern
Saving the proof

We know that we have a tight pattern:

\[ \bullet \rightarrow \bullet \] satisfies \( Q \)
\[ \bullet \rightarrow \bullet \] violates \( Q \)

but

Consider its iterates for each \( n \in \mathbb{N} \):

\[ \left( \bullet \rightarrow \bullet \leftarrow \bullet \right)^n \rightarrow \bullet \] satisfies \( Q \) for all \( n \in \mathbb{N} \)
\[ \left( \bullet \rightarrow \bullet \leftarrow \bullet \right)^i \] satisfies \( Q \)
\[ \left( \bullet \rightarrow \bullet \leftarrow \bullet \right)^{i+1} \] violates \( Q \)

→ Reduce from \( \text{PQE}(Q_0) \) as we explained

Case 1: some iterate violates the query:

Case 2: all iterates satisfy the query:
Saving the proof

We know that we have a **tight pattern**:

- satisfies $Q$
- violates $Q$

Consider its **iterates** for each $n \in \mathbb{N}$:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^n \rightarrow \bullet$

**Case 1:** some iterate **violates** the query:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^i \rightarrow \bullet$
  - satisfies $Q$
  - violates $Q$

  $\rightarrow$ Reduce from $\text{PQE}(Q_0)$ as we explained

**Case 2:** all iterates **satisfy** the query:

- $(\bullet \rightarrow \bullet \leftarrow \bullet)^n \rightarrow \bullet$
  - satisfies $Q$ for all $n \in \mathbb{N}$
  - violates $Q$

$\rightarrow$ Call this an **iterable pattern**
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**: 

\[
\begin{array}{c}
\bullet \rightarrow \bullet \leftarrow \bullet^n \\
\end{array}
\]

satisfies \( Q \) for all \( n \in \mathbb{N} \)

but

\[
\begin{array}{c}
\bullet \rightarrow \bullet \\
\end{array}
\]

violates \( Q \)
Using iterable patterns to show hardness of PQE

We have an iterable pattern:

\[ (\bullet \rightarrow \bullet \leftrightarrow \bullet)^n \rightarrow \bullet \]

satisfies \( Q \) for all \( n \in \mathbb{N} \)

but

\[ \bullet \rightarrow \bullet \rightarrow \bullet \]

violates \( Q \)

Idea: reduce from the \( \textbf{P-hard} \) problem source-to-target connectivity:

- Input: undirected graph with a source \( s \) and target \( t \), all edges have probability \( \frac{1}{2} \)
- Output: what is the probability that the source and target are connected?
Using iterable patterns to show hardness of PQE

We have an iterable pattern: \((\bullet \to \bullet \leftrightarrow \bullet)^n \to \bullet\) satisfies \(Q\) for all \(n \in \mathbb{N}\) but \(\bullet \leftrightarrow \bullet \to \bullet\) violates \(Q\).

**Idea:** reduce from the \#P-hard problem *source-to-target connectivity*:

- **Input:** undirected graph with a *source* \(s\) and *target* \(t\), all edges have probability \(1/2\)
- **Output:** what is the *probability* that the source and target are *connected*?

\[\begin{array}{c}
\text{s} \\
\downarrow^{1/2} \\
\text{u} \\
\downarrow^{1/2} \\
\text{t}
\end{array}\]
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:

\[(\bullet \rightarrow \bullet \leftrightarrow \bullet)^n \rightarrow \bullet\]

satisfies \(Q\) for all \(n \in \mathbb{N}\) but

\[
\begin{align*}
\bullet &\quad \bullet \\
\text{violates } Q
\end{align*}
\]

**Idea:** reduce from the **#P-hard** problem **source-to-target connectivity**:

- **Input:** undirected graph with a source \(s\) and target \(t\), all edges have probability \(1/2\)
- **Output:** what is the **probability** that the source and target are **connected**?

\[
\begin{align*}
s &\quad 1/2 & u &\quad 1/2 & t \\
&\quad 1/2 & &\quad 1/2 &
\end{align*}
\]

is coded as
Using iterable patterns to show hardness of PQE

We have an iterable pattern:

$$\underbrace{(\bullet \rightarrow \bullet \leftarrow \bullet)^n}_s \quad \text{satisfies } Q \quad \text{for all } n \in \mathbb{N}$$

but

$$\underbrace{\bullet \quad \bullet \quad \bullet}_n \quad \text{violates } Q$$

Idea: reduce from the \#P-hard problem source-to-target connectivity:

- Input: undirected graph with a source \(s\) and target \(t\), all edges have probability \(1/2\)
- Output: what is the probability that the source and target are connected?

\[
\begin{array}{c}
\text{s} \\
\text{1/2} \\
\text{u} \\
\text{1/2} \\
\text{t} \\
\text{1/2}
\end{array}
\]

is coded as

\[
\begin{array}{c}
\text{1/2} \\
\text{1/2} \\
\text{1/2}
\end{array}
\]
Using iterable patterns to show hardness of PQE

We have an iterable pattern:

\[(\bullet \rightarrow \bullet \leftarrow \bullet)^n \rightarrow \bullet\]

satisfies $Q$ for all $n \in \mathbb{N}$

but

\[\bullet \rightarrow \bullet \leftarrow \bullet\]

violates $Q$

Idea: reduce from the \textbf{#P-hard} problem \textit{source-to-target connectivity}:

- Input: undirected graph with a source $s$ and target $t$, all edges have probability $1/2$
- Output: what is the probability that the source and target are connected?

\[\begin{array}{c}
s \quad 1/2 \quad u \quad 1/2 \\
\quad 1/2 \quad t
\end{array}\]

is coded as

\[\begin{array}{c}
\bullet \rightarrow \bullet \leftarrow \bullet \\
1/2 \\
\bullet \rightarrow \bullet \leftarrow \bullet
\end{array}\]

Idea: There is a path connecting $s$ and $t$ in a possible world of the graph at the left iff the query $Q$ is satisfied in the corresponding possible world of the TID at the right.
Using iterable patterns to show hardness of PQE

We have an iterable pattern:

\[
\left( \bullet \rightarrow \bullet \leftarrow \bullet \right)^n \rightarrow \bullet
\]

satisfies \( Q \) for all \( n \in \mathbb{N} \)

but

\[
\text{violates } Q
\]

Idea: reduce from the \#P-hard problem source-to-target connectivity:

- Input: undirected graph with a source \( s \) and target \( t \), all edges have probability \( 1/2 \)
- Output: what is the probability that the source and target are connected?

\[
\begin{array}{c}
\text{s} \\
\hline
u \\
\hline
\text{t}
\end{array}
\]

is coded as

Idea: There is a path connecting \( s \) and \( t \) in a possible world of the graph at the left if and only if the query \( Q \) is satisfied in the corresponding possible world of the TID at the right.
Using iterable patterns to show hardness of PQE

We have an iterable pattern:

$$\left( \bullet \rightarrow \bullet \leftarrow \bullet \right)^n \rightarrow \bullet$$

satisfies $Q$ for all $n \in \mathbb{N}$

but

 violates $Q$

Idea: reduce from the $\mathbb{P}$-hard problem source-to-target connectivity:

- Input: undirected graph with a source $s$ and target $t$, all edges have probability $1/2$
- Output: what is the probability that the source and target are connected?

\[ u \]

\[ s \]

\[ t \]

is coded as

Idea: There is a path connecting $s$ and $t$ in a possible world of the graph at the left iff the query $Q$ is satisfied in the corresponding possible world of the TID at the right.
Conclusion and open problems
Conclusion and open problems

- Our result: \( \text{PQE}(Q) \) is \#P-hard for any query \( Q \) closed under homomorphisms unless it is equivalent to a safe UCQ
  - Dichotomy for probabilistic query evaluation over homomorphism-closed queries
  - Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc. (unless they are equivalent to a safe UCQ)

\[ \text{Dichotomy for probabilistic query evaluation over homomorphism-closed queries} \]

- Open problems:
  - The result only applies to graphs, not higher-arity databases
  - We conjecture that the same result holds for higher-arity queries and TIDs
    - Instance transformations are harder to visualize and do not seem to work as-is
  - Does the result still hold for unweighted \( \text{PQE}, \) where all probabilities are one/two?
    - \( \text{PQE} \) for non-hierarchical self-join-free CQs was recently shown to be \#P-hard in this sense
    - [Amarilli and Kimelfeld, two zero]
    - Similar techniques may adapt for our work, but not to the unsafe UCQs...

Thanks for your attention!
Conclusion and open problems

• Our result: \( \text{PQE}(Q) \) is \#P-hard for any query \( Q \) closed under homomorphisms unless it is equivalent to a safe UCQ
  → Dichotomy for probabilistic query evaluation over \text{homomorphism-closed} queries
  → Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc. (unless they are equivalent to a safe UCQ)

• Open problems:
  • The result only applies to graphs, not higher-arity databases
Conclusion and open problems

• Our result: \( \text{PQE}(Q) \) is \#P-hard for any query \( Q \) closed under homomorphisms unless it is equivalent to a safe UCQ
  → Dichotomy for probabilistic query evaluation over \text{homomorphism-closed} queries
  → Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc.
    (unless they are equivalent to a safe UCQ)

• Open problems:
  • The result only applies to \textbf{graphs}, not higher-arity databases
    • We \textit{conjecture} that the same result holds for higher-arity queries and TIDs
    • But instance transformations are \textbf{harder to visualize} and do not seem to work as-is
Our result: \( \text{PQE}(Q) \) is \#P-hard for any query \( Q \) closed under homomorphisms unless it is equivalent to a safe UCQ

\[
\begin{align*}
\rightarrow & \quad \text{Dichotomy for probabilistic query evaluation over homomorphism-closed queries} \\
\rightarrow & \quad \text{Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc.} \\
& \quad \text{unless they are equivalent to a safe UCQ}
\end{align*}
\]

Open problems:

- The result only applies to graphs, not higher-arity databases
  - We conjecture that the same result holds for higher-arity queries and TIDs
  - But instance transformations are harder to visualize and do not seem to work as-is

- Does the result still hold for unweighted PQE, where all probabilities are \( 1/2 \)?
Conclusion and open problems

- Our result: \( \text{PQE}(Q) \) is \#P-hard for any query \( Q \) closed under homomorphisms unless it is equivalent to a safe UCQ
  - Dichotomy for probabilistic query evaluation over \textit{homomorphism-closed} queries
  - Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc. (unless they are equivalent to a safe UCQ)

- Open problems:
  - The result only applies to graphs, not higher-arity databases
    - We \textit{conjecture} that the same result holds for higher-arity queries and TIDs
    - But instance transformations are \textit{harder to visualize} and do not seem to work as-is
  - Does the result still hold for \textit{unweighted} PQE, where all probabilities are \( 1/2 \)?
    - PQE for \textit{non-hierarchical self-join-free} CQs was recently shown to be \#P-hard in this sense [Amarilli and Kimelfeld, 2020]
    - Similar techniques \textit{may adapt} for our work, but not to the unsafe UCQs...
Conclusion and open problems

- Our result: $\text{PQE}(Q)$ is \textbf{#P-hard} for any query $Q$ closed under homomorphisms unless it is equivalent to a safe UCQ
  - Dichotomy for probabilistic query evaluation over \textit{homomorphism-closed} queries
  - Implies intractability for RPQs, Datalog queries, ontology-mediated queries, etc.
    (unless they are equivalent to a safe UCQ)

- \textbf{Open problems:}
  - The result only applies to graphs, not higher-arity databases
    - We conjecture that the same result holds for higher-arity queries and TIDs
    - But instance transformations are harder to visualize and do not seem to work as-is
  - Does the result still hold for \textit{unweighted} PQE, where all probabilities are $1/2$?
    - PQE for \textit{non-hierarchical self-join-free CQs} was recently shown to be \textbf{#P-hard} in this sense
      [Amarilli and Kimelfeld, 2020]
    - Similar techniques \textit{may adapt} for our work, but not to the unsafe UCQs...

Thanks for your attention!
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:

![Diagram]

If $Q$ becomes false at one step, then we have found a tight pattern. Otherwise, we have found a contradiction:

- The disconnection process terminates
- At the end of the process, we obtain a union of stars $D'$
- It is homomorphically equivalent to a constant-sized $D''$ satisfying $Q$
- $D''$ has a homomorphism back to $D$
- This contradicts the minimality of the large $D$
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:

  ![Diagram showing the process of disconnecting edges in a minimal model](diagram.png)

  - If $Q$ becomes false at one step, then we have found a tight pattern
  - Otherwise, we have found a contradiction:
    - The disconnection process terminates
    - At the end of the process, we obtain a union of stars $D'$
    - It is homomorphically equivalent to a constant-sized $D''$ satisfying $Q$
    - $D''$ has a homomorphism back to $D$
    - This contradicts the minimality of the large $D$
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:

  ![Diagram showing the disconnection process](image)

- If $Q$ becomes false at one step, then we have found a **tight pattern**
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models.
- Take a large minimal model $D$ and **disconnect its edges**:

  ![Diagram](image)

  - If $Q$ becomes false at one step, then we have found a **tight pattern**.
  - Otherwise, we have found a **contradiction**:
    - The disconnection process **terminates**.
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models.
- Take a large minimal model $D$ and **disconnect its edges**:

![Diagram showing the process of disconnecting edges and obtaining a union of stars]

- If $Q$ becomes false at one step, then we have found a **tight pattern**.
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**.
  - At the end of the process, we obtain a **union of stars** $D'$.
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models.
- Take a large minimal model $D$ and **disconnect its edges**:

![Disconnected graph example]

- If $Q$ becomes false at one step, then we have found a **tight pattern**.
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**.
  - At the end of the process, we obtain a **union of stars** $D'$.
  - It is **homomorphically equivalent** to a constant-sized $D''$ satisfying $Q$. 
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model $D$ and **disconnect its edges**:

  ![Diagram]

  - If $Q$ becomes false at one step, then we have found a **tight pattern**
  - Otherwise, we have found a **contradiction**:
    - The disconnection process **terminates**
    - At the end of the process, we obtain a **union of stars** $D'$
    - It is **homomorphically equivalent** to a constant-sized $D''$ satisfying $Q$
    - $D''$ has a **homomorphism** back to $D$
Why can we always find tight patterns?

- Unbounded queries have *arbitrarily large* minimal models
- Take a large minimal model $D$ and *disconnect its edges*:

- If $Q$ becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **union of stars** $D'$
  - It is **homomorphically equivalent** to a constant-sized $D''$ satisfying $Q$
  - $D''$ has a **homomorphism** back to $D$
  - This contradicts the **minimality** of the large $D$
How to show \#P-hardness for PQE

How to show the \#P-hardness of PQE for the unsafe query $Q : x \rightarrow y \rightarrow z \rightarrow w$
How to show \#P-hardness for PQE

How to show the \textbf{\#P-hardness} of PQE for the \textit{unsafe} query $Q : x \rightarrow y \rightarrow z \rightarrow w$

- Reduce from the problem of \textit{counting satisfying valuations} of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
How to show \#P-hardness for PQE

How to show the **\#P-hardness** of PQE for the **unsafe** query $Q : x \rightarrow y \rightarrow z \rightarrow w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already **\#P-hard** for so-called **PP2DNF formulas**:
How to show \( \#P \)-hardness for PQE

How to show the \( \#P \)-hardness of PQE for the unsafe query \( Q : x \rightarrow y \rightarrow z \rightarrow w \)

- Reduce from the problem of \textit{counting satisfying valuations} of a Boolean formula
  - e.g., given \((x \lor y) \land z\), compute that it has 3 satisfying valuations
- This problem is already \( \#P \)-hard for so-called \texttt{PP2DNF} formulas:
  - \textbf{Positive} (no negation) and \textbf{Partitioned variables}: \( X_1, \ldots, X_n \) and \( Y_1, \ldots, Y_m \)
How to show \#P-hardness for PQE

How to show the \#P-hardness of PQE for the unsafe query $Q : x \rightarrow y \rightarrow z \rightarrow w$

- Reduce from the problem of counting satisfying valuations of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already \#P-hard for so-called PP2DNF formulas:
  - Positive (no negation) and Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \land Y_j$
How to show \( \#P \)-hardness for PQE

How to show the \( \#P \)-hardness of PQE for the unsafe query \( Q: x \rightarrow y \rightarrow z \rightarrow w \)

- Reduce from the problem of counting satisfying valuations of a Boolean formula
  - e.g., given \((x \lor y) \land z\), compute that it has 3 satisfying valuations
- This problem is already \( \#P \)-hard for so-called PP2DNF formulas:
  - Positive (no negation) and Partitioned variables: \( X_1, \ldots, X_n \) and \( Y_1, \ldots, Y_m \)
  - 2-DNF: disjunction of clauses like \( X_i \land Y_j \)
- Example: \( \phi: (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2) \)
How to show \#P-hardness for PQE

How to show the **\#P-hardness** of PQE for the unsafe query \( Q : x \rightarrow y \rightarrow z \rightarrow w \)

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given \( (x \lor y) \land z \), compute that it has 3 satisfying valuations
- This problem is already **\#P-hard** for so-called **PP2DNF formulas**:
  - Positive (no negation) and **Partitioned variables**: \( X_1, \ldots, X_n \) and \( Y_1, \ldots, Y_m \)
  - 2-DNF: disjunction of clauses like \( X_i \land Y_j \)

- Example: \( \phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2) \)

- Example:
  - \( a'_1 \rightarrow a_1 \)
  - \( a'_2 \rightarrow a_2 \)
  - \( a'_3 \rightarrow a_3 \)
How to show \#P-hardness for PQE

How to show the \#P-hardness of PQE for the unsafe query $Q : x \rightarrow y \rightarrow z \rightarrow w$

- Reduce from the problem of counting satisfying valuations of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already \#P-hard for so-called PP2DNF formulas:
  - Positive (no negation) and Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \land Y_j$

- Example: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

$$\begin{align*}
a'_1 & \xrightarrow{1/2} a_1 \\
\color{red}{a'_2} & \xrightarrow{1/2} a_2 \\
\color{red}{a'_3} & \xrightarrow{1/2} a_3 \\
\end{align*} \quad \quad \quad \quad \quad \quad 
\begin{align*}
\color{red}{b'_1} & \xrightarrow{1/2} b_1 \\
b_2 & \xrightarrow{1/2} b_2 \\
\end{align*}$$

Idea: Satisfying valuations of $\phi$ correspond to possible worlds with a match of $Q$
How to show \#P-hardness for PQE

How to show the \#P-hardness of PQE for the unsafe query \( Q : x \rightarrow y \rightarrow z \rightarrow w \)

- Reduce from the problem of counting satisfying valuations of a Boolean formula
  - e.g., given \((x \lor y) \land z\), compute that it has 3 satisfying valuations
- This problem is already \#P-hard for so-called PP2DNF formulas:
  - Positive (no negation) and Partitioned variables: \( X_1, \ldots, X_n \) and \( Y_1, \ldots, Y_m \)
  - 2-DNF: disjunction of clauses like \( X_i \land Y_j \)

- Example: \( \phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2) \)
How to show \#P-hardness for PQE

How to show the \#P-hardness of PQE for the unsafe query $Q: x \rightarrow y \rightarrow z \rightarrow w$

- Reduce from the problem of counting satisfying valuations of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already \#P-hard for so-called PP2DNF formulas:
  - Positive (no negation) and Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - 2-DNF: disjunction of clauses like $X_i \land Y_j$
- Example: $\phi: (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

Idea: Satisfying valuations of $\phi$ correspond to possible worlds with a match of $Q$


**Ontology-based access to probabilistic data with OWL QL.**