Query Evaluation on Probabilistic Data: New Hard Cases

Antoine Amarilli[1], joint work with Benny Kimelfeld[2], İsmail İlkan Ceylan[3]

October 10, 2019

[1]Télécom Paris

[2]Technion

[3]University of Oxford

Uncertain data management

- Databases: manage data and answer queries over it

Uncertain data management

- **Databases:** manage **data** and answer **queries** over it

WorksAt	
Antoine	Télécom Paris
Antoine	Paris Sud
Benny	Paris Sud
Benny	Technion
İsmail	U. Oxford

Uncertain data management

- **Databases:** manage **data** and answer **queries** over it

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

Uncertain data management

- Databases: manage data and answer queries over it
- In this talk, data is simply a labeled graph

WorksAt	
Antoine	Télécom Paris
Antoine	Paris Sud
Benny	Paris Sud
Benny	Technion
İsmail	U. Oxford

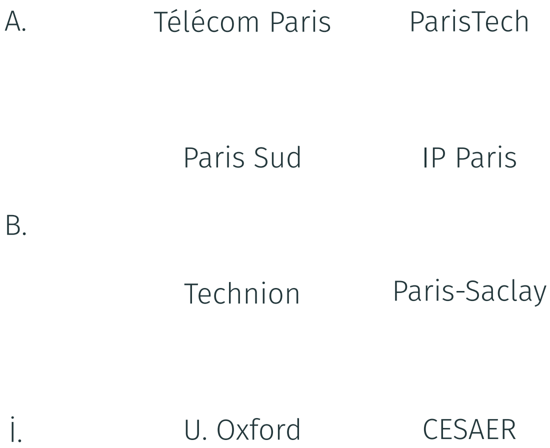
MemberOf	
Télécom Paris	ParisTech
Télécom Paris	IP Paris
Paris Sud	IP Paris
Paris Sud	Paris-Saclay
Technion	CESAER

Uncertain data management

- **Databases:** manage **data** and answer **queries** over it
- In this talk, **data** is simply 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

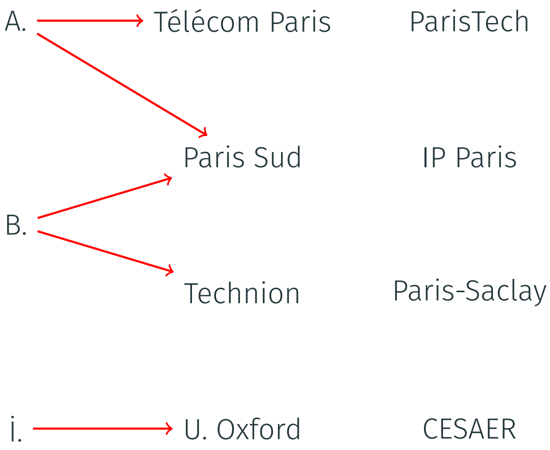


Uncertain data management

- **Databases:** manage **data** and answer **queries** over it
- In this talk, **data** is simply 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

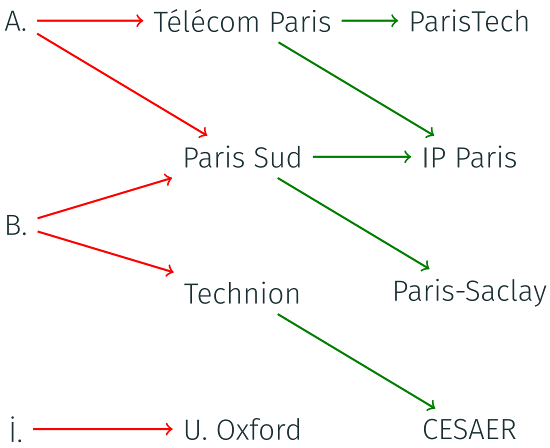


Uncertain data management

- **Databases:** manage **data** and answer **queries** over it
- In this talk, **data** is simply 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

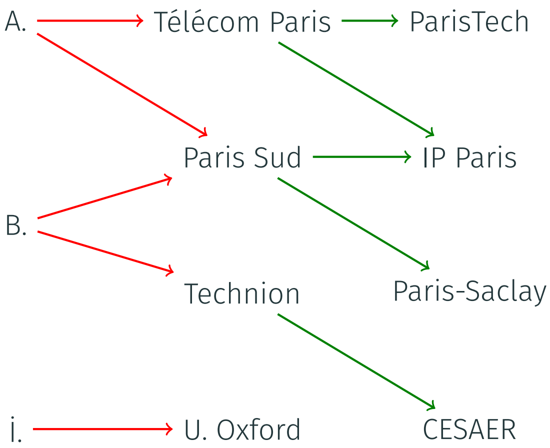


Uncertain data management

- **Databases:** manage **data** and answer **queries** over it
- In this talk, **data** is simply a labeled graph

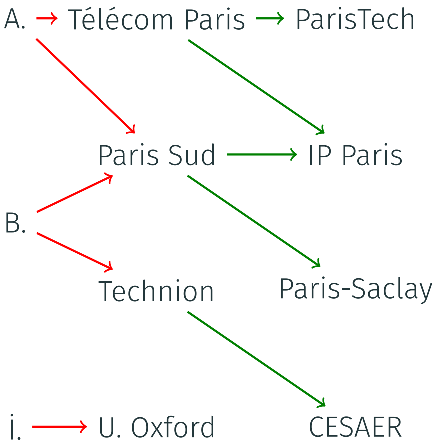
WorksAt	
Antoine	Télécom Paris
Antoine	Paris Sud
Benny	Paris Sud
Benny	Technion
İsmail	U. Oxford

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



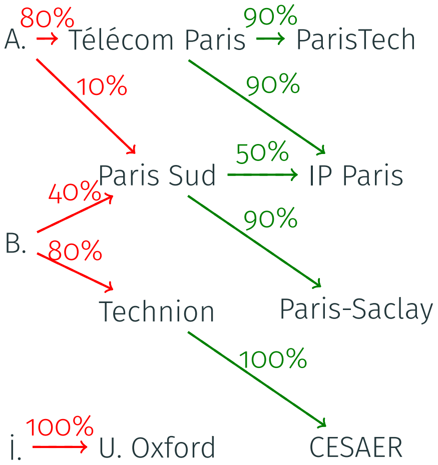
→ **Problem:** we may be **uncertain** about the data

Uncertain data model



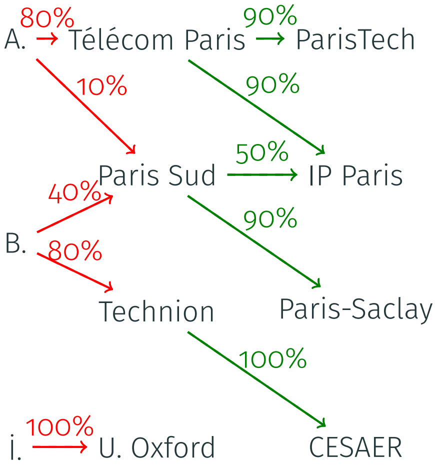
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**

Uncertain data model



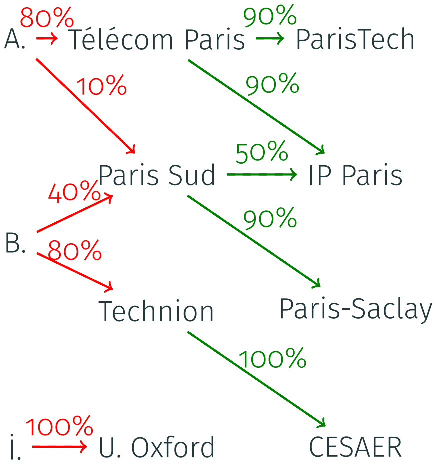
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**

Uncertain data model



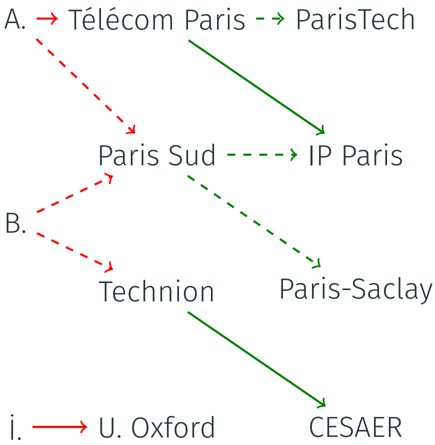
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**

Uncertain data model



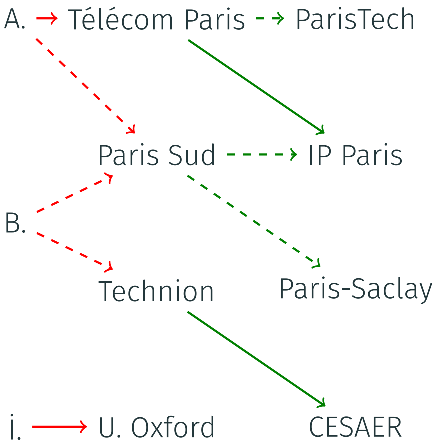
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts

Uncertain data model



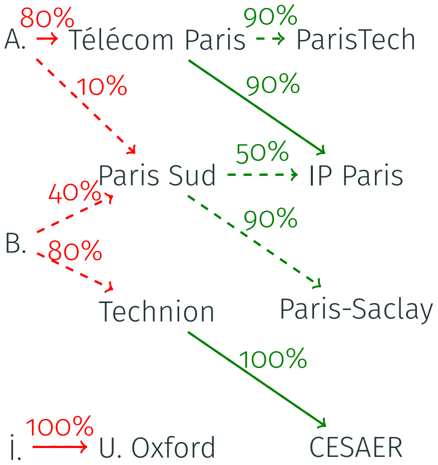
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts

Uncertain data model



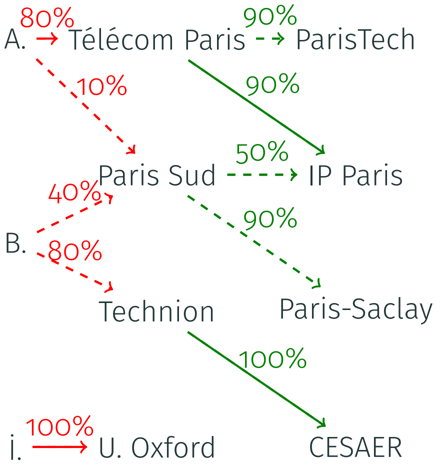
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts
- What is **probability** of this possible world?

Uncertain data model



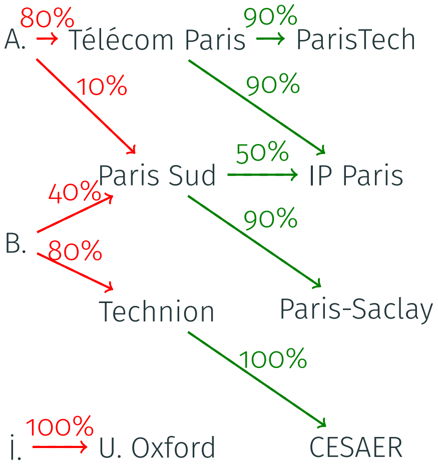
- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts
- What is **probability** of this possible world?

Uncertain data model



- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts
- What is **probability** of this possible world? **0.03%**

Uncertain data model



- Uncertain data model: **TID**, for **tuple-independent database**
- Every fact carries a **probability**
- Every fact exists with the indicated **probability**
- All facts are **independent**
- **Possible world:** subset of facts
- What is **probability** of this possible world? **0.03%**

→ This model is **simplistic**, but already challenging to understand

Queries

- Query: maps a non-probabilistic graph to YES/NO

Queries

- **Query:** maps a **non-probabilistic** graph to YES/NO
- **Conjunctive query:** can I find an occurrence of a **pattern**?

  $x \longrightarrow y \longrightarrow z$

Queries

- **Query:** maps a non-probabilistic graph to YES/NO
- **Conjunctive query:** can I find an occurrence of a pattern?

  $x \longrightarrow y \longrightarrow z$

  - We want a homomorphism from the pattern to the graph

Queries

- **Query:** maps a non-probabilistic graph to YES/NO
- **Conjunctive query:** can I find an occurrence of a pattern?

  $x \longrightarrow y \longrightarrow z$

  - We want a homomorphism from the pattern to the graph
  - Not necessarily injective!

Queries

- Query: maps a non-probabilistic graph to YES/NO
- Conjunctive query: can I find an occurrence of a pattern?

  $x \longrightarrow y \longrightarrow z$

  - We want a homomorphism from the pattern to the graph
  - Not necessarily injective!
- Union of conjunctive queries: does one of the patterns match?

Queries

- **Query:** maps a **non-probabilistic** graph to YES/NO
- **Conjunctive query:** can I find an occurrence of a **pattern**?

  $$x \longrightarrow y \longrightarrow z$$

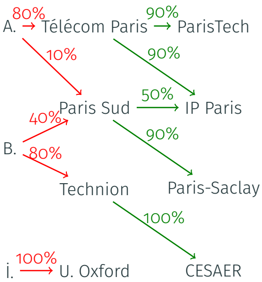
  - We want a **homomorphism** from the pattern to the graph
  - Not necessarily **injective**!
- **Union of conjunctive queries**: does one of the patterns match?
- **Homomorphism-closed query Q**: if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q

Problem statement: Probabilistic query evaluation (PQE)

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

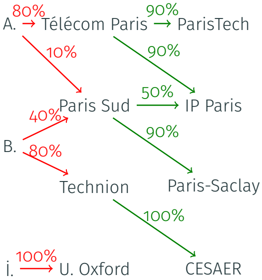
Problem statement: Probabilistic query evaluation (PQE)

- We fix a query Q, for instance: $x \longrightarrow y \longrightarrow z$
- The input is a TID:



Problem statement: Probabilistic query evaluation (PQE)

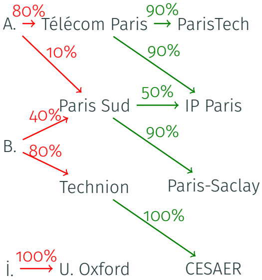
- We **fix** a query Q, for instance: $x \longrightarrow y \longrightarrow z$
- The **input** is a TID:



- 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: $x \longrightarrow y \longrightarrow z$
- The **input** is a TID:

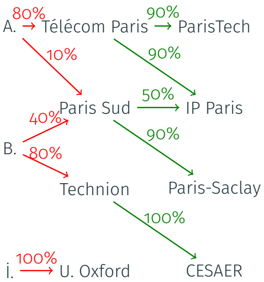


- The **output** is the **total probability** of the worlds which satisfy the query
  - → **Intuition:** the **probability** that the query is true

Problem statement: Probabilistic query evaluation (PQE)

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

- The **input** is a TID:



- The **output** is the **total probability** of the worlds which satisfy the query
  - → **Intuition:** the **probability** that the query is true

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

Existing results

Dichotomy on the unions of conjunctive queries (UCQs):

Theorem [Dalvi and Suciu, 2012]

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

Existing results

Dichotomy on the unions of conjunctive queries (UCQs):

Theorem [Dalvi and Suciu, 2012]

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

- Our example query $x \longrightarrow y \longrightarrow z$ is...

Existing results

Dichotomy on the unions of conjunctive queries (UCQs):

Theorem [Dalvi and Suciu, 2012]

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

- Our example query $x \longrightarrow y \longrightarrow z$ is... safe

Existing results

Dichotomy on the unions of conjunctive queries (UCQs):

Theorem [Dalvi and Suciu, 2012]

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

- Our example query $x \longrightarrow y \longrightarrow z$ is... safe

Also: dichotomy on the instance families:

Theorem [Amarilli et al., 2015, Amarilli et al., 2016]

- *For any query* Q *in monadic second-order logic,* $\mathrm{PQE}(Q)$ *is in PTIME if the input TIDs have bounded treewidth*

Existing results

Dichotomy on the unions of conjunctive queries (UCQs):

Theorem [Dalvi and Suciu, 2012]

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

- Our example query $x \longrightarrow y \longrightarrow z$ is... safe

Also: dichotomy on the instance families:

Theorem [Amarilli et al., 2015, Amarilli et al., 2016]

- *For any query Q in monadic second-order logic, $\mathrm{PQE}(Q)$ is in PTIME if the input TIDs have bounded treewidth*
- *There is a query Q such that $\mathrm{PQE}(Q)$ is #P-hard on any TID family of unbounded treewidth (with several technical assumptions)*

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - Positive: no negation

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - Positive: no negation
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - Positive: no negation
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$

  $\rightarrow$ #SAT is already #P-hard

Why are some queries unsafe?

This query is unsafe: $x \xrightarrow{\quad\quad} y \xrightarrow{\quad\quad} z \xrightarrow{\quad\quad} w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - Positive: no negation
  - 2-DNF: disjunction of clauses like $X_i \land Y_j$

  $\rightarrow$ #SAT is already #P-hard
- Example: $(X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

Why are some queries unsafe?

This query is unsafe: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- #SAT: counting satisfying valuations of a Boolean formula
- Specifically, reduce from #PP2DNF:
  - Partitioned variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - Positive: no negation
  - 2-DNF: disjunction of clauses like $X_i \wedge Y_j$

  $\rightarrow$ #SAT is already #P-hard
- Example: $(X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$$a'_1 \xrightarrow{1/2} a_1$$

$$a'_2 \xrightarrow{1/2} a_2$$

$$a'_3 \xrightarrow{1/2} a_3$$

Why are some queries unsafe?

This query is **unsafe**: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- **#SAT**: counting satisfying valuations of a Boolean formula
- Specifically, reduce from **#PP2DNF**:
  - **Partitioned** variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - **Positive**: no negation
  - **2-DNF**: disjunction of clauses like $X_i \wedge Y_j$

  $\rightarrow$ **#SAT** is already **#P-hard**
- Example: $(X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

$a'_1 \xrightarrow{1/2} a_1$ $\qquad$ $b_1 \xrightarrow{1/2} b'_1$

$a'_2 \xrightarrow{1/2} a_2$

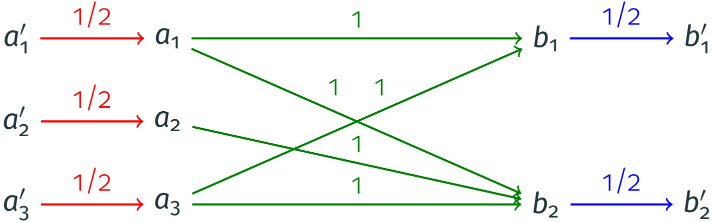
$a'_3 \xrightarrow{1/2} a_3$ $\qquad$ $b_2 \xrightarrow{1/2} b'_2$

Why are some queries unsafe?

This query is **unsafe**: $x \longrightarrow y \longrightarrow z \longrightarrow w$

- **#SAT**: counting satisfying valuations of a Boolean formula
- Specifically, reduce from **#PP2DNF**:
  - **Partitioned** variables: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - **Positive**: no negation
  - **2-DNF**: disjunction of clauses like $X_i \land Y_j$

  $\rightarrow$ **#SAT** is already **#P-hard**
- Example: $(X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$



New results in this talk

We present more cases where PQE is #P-hard:

- With İsmail İlkan Ceylan, for expressive queries:

Theorem [Amarilli and Ceylan, 2019]

For any query Q closed under homomorphisms, $\mathrm{PQE}(Q)$ *is #P-hard unless Q is equivalent to a safe UCQ*

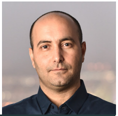
New results in this talk

We present more cases where PQE is #P-hard:

- With İsmail İlkan Ceylan, for expressive queries:

Theorem [Amarilli and Ceylan, 2019]

For any query Q closed under homomorphisms, $\mathrm{PQE}(Q)$ *is #P-hard unless* Q *is equivalent to a safe UCQ*



- With Benny Kimelfeld, in the unweighted case:

Theorem [Amarilli and Kimelfeld, 2019]

For any CQ Q without self-joins (every edge has a different color), if Q *is unsafe then* $\mathrm{PQE}(Q)$ *is #P-hard even if all probabilities are* $1/2$

Hardness for queries closed under homomorphisms

Result statement

We consider queries closed under homomorphisms:

Result statement

We consider queries closed under homomorphisms:

- Generalizes CQs and UCQs, Datalog, Regular path queries...
  - Example: WorksAt/MemberOf$^+$

Result statement

We consider queries closed under homomorphisms:

- Generalizes CQs and UCQs, Datalog, Regular path queries...
  - Example: WorksAt/MemberOf$^+$
- Equivalent phrasing: infinite union of CQs
  - Example: WA/MO, WA/MO/MO, WA/MO/MO/MO, ...

Result statement

We consider queries closed under homomorphisms:

- Generalizes CQs and UCQs, Datalog, Regular path queries...
  - Example: WorksAt/MemberOf$^+$
- Equivalent phrasing: infinite union of CQs
  - Example: WA/MO, WA/MO/MO, WA/MO/MO/MO, ...

Theorem

- *Some queries closed under homomorphisms are UCQs: the previous dichotomy applies, PQE is PTIME or #P-hard*
- *For all other queries closed under homomorphisms, PQE is #P-hard*

Result statement

We consider queries **closed under homomorphisms**:

- Generalizes **CQs** and **UCQs**, **Datalog**, **Regular path queries**...
  - Example: WorksAt/MemberOf$^+$
- Equivalent phrasing: **infinite** union of CQs
  - Example: WA/MO, WA/MO/MO, WA/MO/MO/MO, ...

Theorem

- *Some queries closed under homomorphisms are UCQs: the previous dichotomy applies, PQE is PTIME or #P-hard*
- *For all other queries closed under homomorphisms, PQE is #P-hard*

- The query WA/MO$^+$ is **equivalent** to WA/MO which is a **safe UCQ**

Result statement

We consider queries **closed under homomorphisms**:

- Generalizes **CQs** and **UCQs**, **Datalog**, **Regular path queries**...
  - Example: WorksAt/MemberOf$^+$
- Equivalent phrasing: **infinite** union of CQs
  - Example: WA/MO, WA/MO/MO, WA/MO/MO/MO, ...

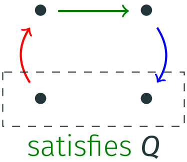
Theorem

- *Some queries closed under homomorphisms are UCQs: the previous dichotomy applies, PQE is PTIME or #P-hard*
- *For all other queries closed under homomorphisms, PQE is #P-hard*

- The query WA/MO$^+$ is **equivalent** to WA/MO which is a **safe UCQ**
- The query WA/MO$^+$/IN is **not equivalent to a UCQ** so PQE is **#P-hard**

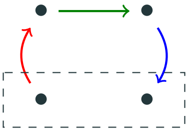
Proof idea: finding hard patterns

- Fix the query Q and find a tight pattern, i.e,. a graph such that:



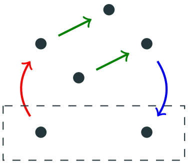
Proof idea: finding hard patterns

- Fix the query Q and find a tight pattern, i.e,. a graph such that:



satisfies Q

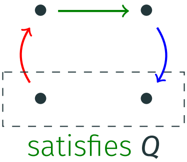
but



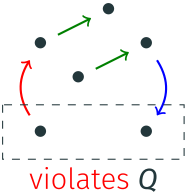
violates Q

Proof idea: finding hard patterns

- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:



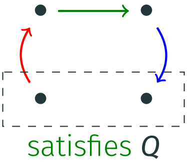
but



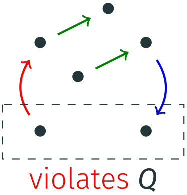
- If the query is **unbounded**, we can find a **tight pattern**

Proof idea: finding hard patterns

- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:



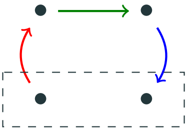
but



- If the query is **unbounded**, we can find a **tight pattern**
  - Unbounded queries have **arbitrarily large** minimal models
  - Take one such model and **disconnect edges** as much as possible

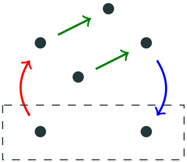
Proof idea: finding hard patterns

- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:



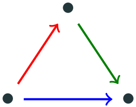
satisfies Q

but



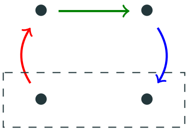
violates Q

- If the query is **unbounded**, we can find a **tight pattern**
  - Unbounded queries have **arbitrarily large** minimal models
  - Take one such model and **disconnect edges** as much as possible
  - Unbounded queries have **arbitrarily large** minimal models



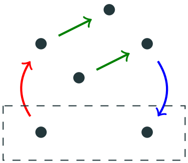
Proof idea: finding hard patterns

- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:



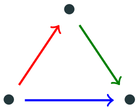
satisfies Q

but

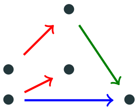


violates Q

- If the query is **unbounded**, we can find a **tight pattern**
  - Unbounded queries have **arbitrarily large** minimal models
  - Take one such model and **disconnect edges** as much as possible
  - Unbounded queries have **arbitrarily large** minimal models

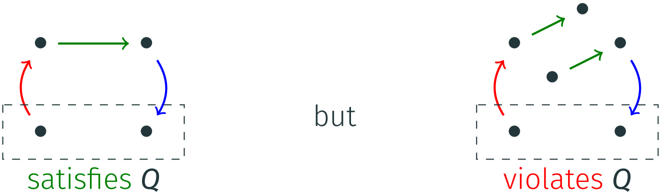


to

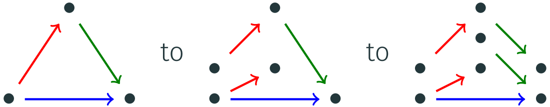


Proof idea: finding hard patterns

- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:

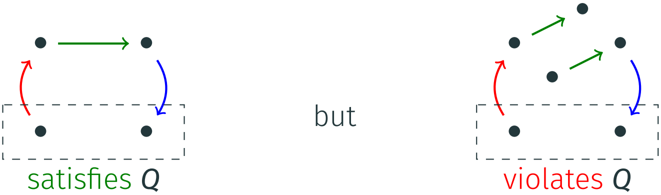


- If the query is **unbounded**, we can find a **tight pattern**
  - Unbounded queries have **arbitrarily large** minimal models
  - Take one such model and **disconnect edges** as much as possible
  - Unbounded queries have **arbitrarily large** minimal models

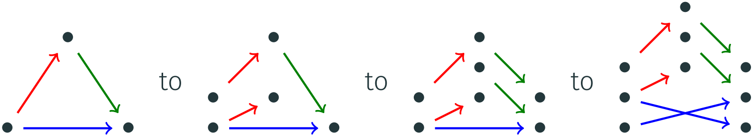


Proof idea: finding hard patterns

- Fix the query Q and find a tight pattern, i.e,. a graph such that:

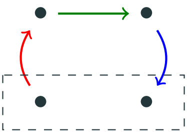


- If the query is unbounded, we can find a tight pattern
  - Unbounded queries have arbitrarily large minimal models
  - Take one such model and disconnect edges as much as possible
  - Unbounded queries have arbitrarily large minimal models

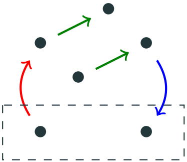


Proof idea: finding hard patterns

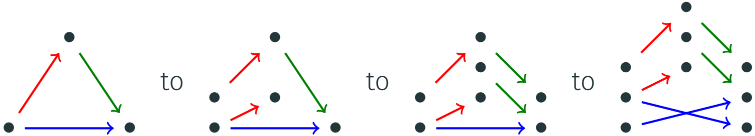
- Fix the query Q and find a **tight pattern**, i.e,. a graph such that:



but

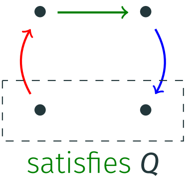


- If the query is **unbounded**, we can find a **tight pattern**
  - Unbounded queries have **arbitrarily large** minimal models
  - Take one such model and **disconnect edges** as much as possible
  - Unbounded queries have **arbitrarily large** minimal models

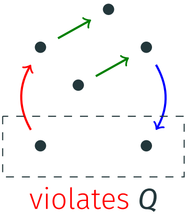


  - If Q is still true then the model is "explained" by a **union of stars**

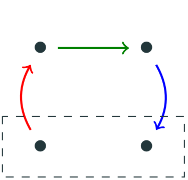
Using hard patterns for #P-hardness



but

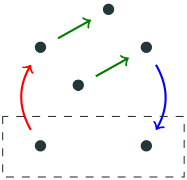


We can reduce from #PP2DNF like before:



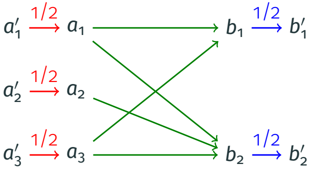
satisfies Q

but

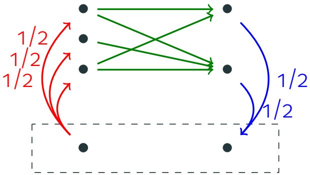


violates Q

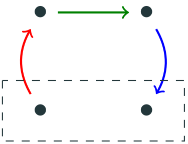
We can reduce from #PP2DNF like before:



is coded as

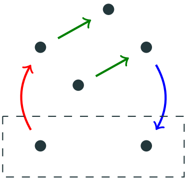


Using hard patterns for #P-hardness



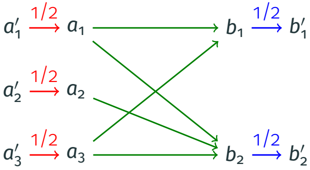
satisfies Q

but

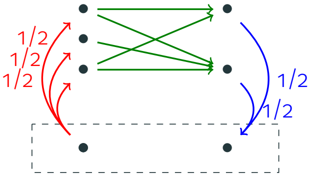


violates Q

We can reduce from #PP2DNF like before:

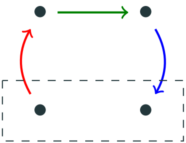


is coded as



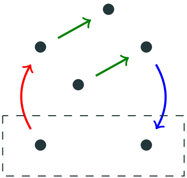
Idea: possible worlds at the **left** have a path $x \longrightarrow y \longrightarrow z \longrightarrow w$
iff the corresponding world at the **right** satisfies Q...

Using hard patterns for #P-hardness



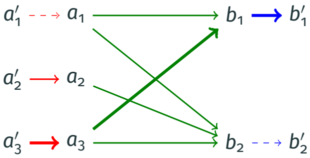
satisfies Q

but

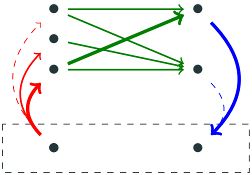


violates Q

We can reduce from #PP2DNF like before:

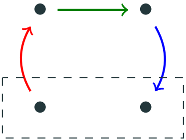


is coded as



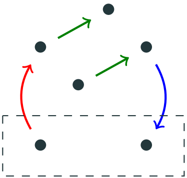
Idea: possible worlds at the **left** have a path $x \longrightarrow y \longrightarrow z \longrightarrow w$ iff the corresponding world at the **right** satisfies Q...

Using hard patterns for #P-hardness



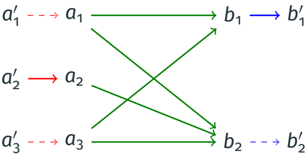
satisfies Q

but

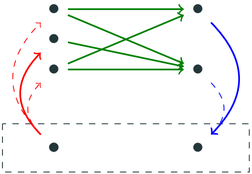


violates Q

We can reduce from #PP2DNF like before:

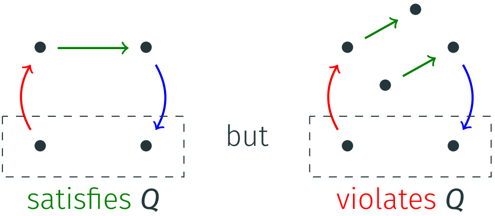


is coded as

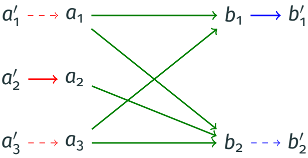


Idea: possible worlds at the **left** have a path $x \longrightarrow y \longrightarrow z \longrightarrow w$ iff the corresponding world at the **right** satisfies Q...

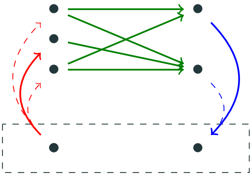
Using hard patterns for #P-hardness



We can reduce from #PP2DNF like before:

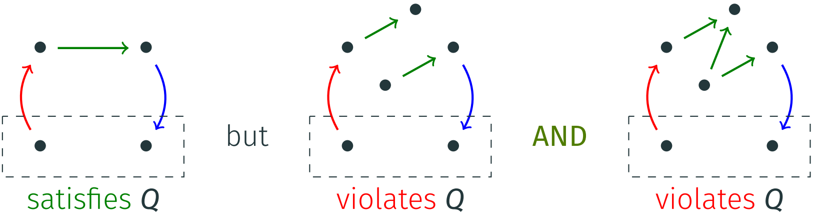


is coded as

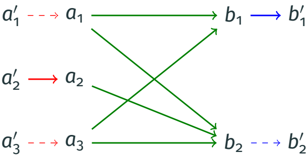


Idea: possible worlds at the **left** have a path $x \longrightarrow y \longrightarrow z \longrightarrow w$
iff the corresponding world at the **right** satisfies Q...
... except we need **more** from the hard pattern!

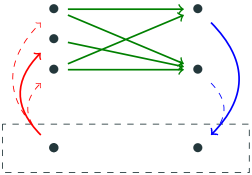
Using hard patterns for #P-hardness



We can reduce from #PP2DNF like before:



is coded as



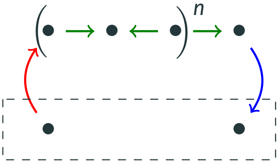
Idea: possible worlds at the left have a path $x \longrightarrow y \longrightarrow z \longrightarrow w$
iff the corresponding world at the right satisfies Q...
... except we need more from the hard pattern!

From tight patterns to iterable patterns

When we cannot find a tight pattern, we can find an iterable pattern:

From tight patterns to iterable patterns

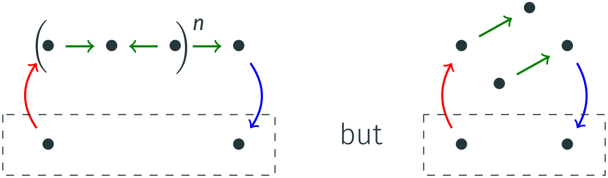
When we cannot find a tight pattern, we can find an iterable pattern:



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

From tight patterns to iterable patterns

When we cannot find a tight pattern, we can find an iterable pattern:

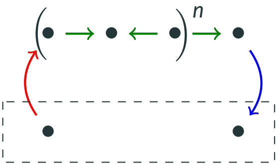


satisfies Q for all $n \in \mathbb{N}$

violates Q

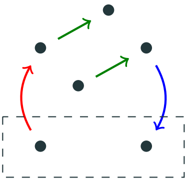
From tight patterns to iterable patterns

When we cannot find a tight pattern, we can find an **iterable pattern**:



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

but



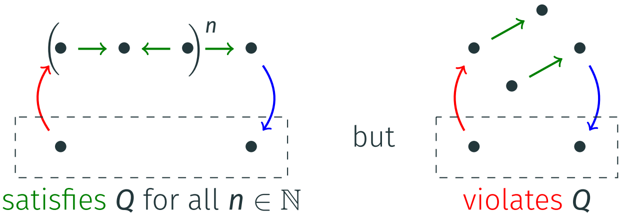
violates Q

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

- Given a TID with a **source** and **sink**, what is the **probability** that the sink is **reachable** from the sink?

From tight patterns to iterable patterns

When we cannot find a tight pattern, we can find an iterable pattern:



Idea: use iterable patterns to reduce from the #P-hard problem source-to-target connectivity:

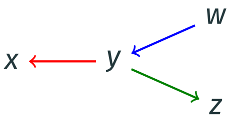
- Given a TID with a source and sink, what is the probability that the sink is reachable from the sink?

Technically challenging to get a correct reduction!

Hardness for unweighted PQE

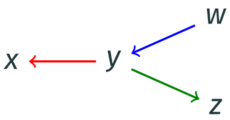
Problem statement

- We restrict back to CQs
- We impose self-join-freeness: every edge color is different



Problem statement

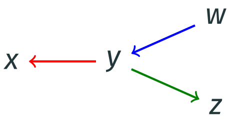
- We restrict back to CQs
- We impose self-join-freeness: every edge color is different



- Existing dichotomy:
  - If Q only consists of stars, then it is safe and $\mathrm{PQE}(Q)$ is in PTIME

Problem statement

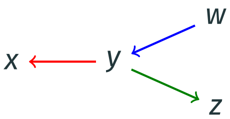
- We restrict back to CQs
- We impose self-join-freeness: every edge color is different



- Existing dichotomy:
  - If Q only consists of stars, then it is safe and $\mathrm{PQE}(Q)$ is in PTIME
  - In all other cases, $\mathrm{PQE}(Q)$ is #P-hard

Problem statement

- We restrict back to CQs
- We impose self-join-freeness: every edge color is different



- Existing dichotomy:
  - If Q only consists of stars, then it is safe and $\mathrm{PQE}(Q)$ is in PTIME
  - In all other cases, $\mathrm{PQE}(Q)$ is #P-hard

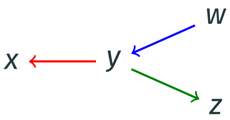
But what if all facts of the TIDs had probability $1/2$?

$\rightarrow$ Equivalently: given a graph G, how many subgraphs satisfy Q

- We call this problem $\mathrm{MC}(Q)$: model counting for Q

Problem statement

- We restrict back to CQs
- We impose **self-join-freeness**: every edge color is different



- Existing dichotomy:
  - If Q only consists of stars, then it is safe and $\mathrm{PQE}(Q)$ is in PTIME
  - In all other cases, $\mathrm{PQE}(Q)$ is #P-hard

But what if all facts of the TIDs had probability $1/2$?

$\rightarrow$ Equivalently: given a graph G, how many subgraphs satisfy Q

- We call this problem $\mathrm{MC}(Q)$: model counting for Q

Theorem

For any self-join-free CQ Q, if Q is unsafe then $MC(Q)$ is #P-hard.

First step: Restricting to a simpler query

For any unsafe query, we can reduce from simpler queries, essentially:

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

$\rightarrow$ We must show that $\mathrm{MC}(Q)$ is #P-hard for this query

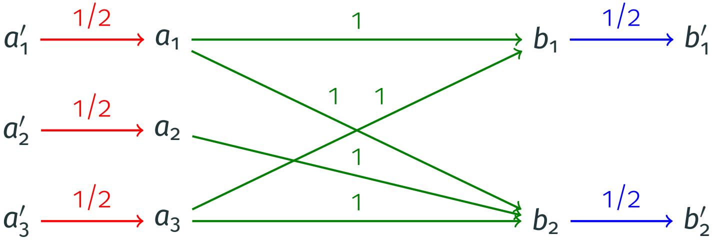
First step: Restricting to a simpler query

For any unsafe query, we can reduce from simpler queries, essentially:

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

$\rightarrow$ We must show that $\mathrm{MC}(Q)$ is #P-hard for this query

Can we use our earlier reduction for #P-hardness of PQE?



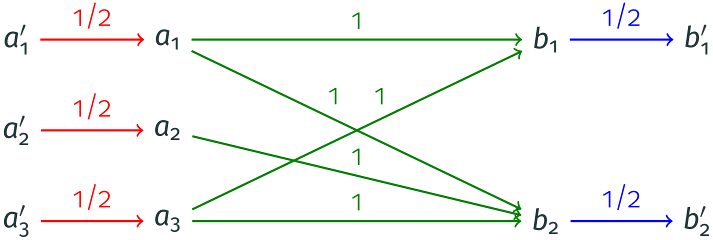
First step: Restricting to a simpler query

For any unsafe query, we can reduce from **simpler queries**, essentially:

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

$\rightarrow$ We must show that $\mathrm{MC}(Q)$ is #P-hard for **this query**

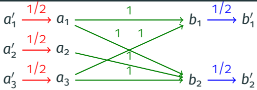
Can we use our **earlier reduction** for #P-hardness of PQE?



$\rightarrow$ Problem: this reduction crucially uses **probability 1**

Getting to an equation system

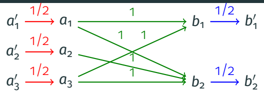
We want to reduce from $\mathrm{PQE}(Q)$, on some graph G with probabilities



Task: count the number X of red-blue edge subsets that violate Q

Getting to an equation system

We want to reduce from $\mathrm{PQE}(Q)$, on some graph G with probabilities

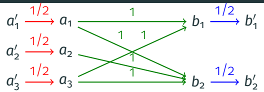


Task: count the number X of red-blue edge subsets that violate Q

- Split the subsets on some parameter e.g., the number of nodes
  $\rightarrow\ X = X_1, \ldots, X_k$

Getting to an equation system

We want to reduce from $\mathrm{PQE}(Q)$, on some graph G with probabilities

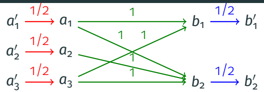


Task: count the number X of red-blue edge subsets that violate Q

- Split the subsets on some parameter e.g., the number of nodes
  $\rightarrow X = X_1, \ldots, X_k$
- Create unweighted copies of G modified with some gadgets e.g., replace each edge by multiple copies of a path
  $\rightarrow$ Created $G_1, \ldots, G_k$
  $\rightarrow$ Call the oracle for $\mathrm{MC(Q)}$ on each to get $N_1, \ldots, N_k$

Getting to an equation system

We want to reduce from $\mathrm{PQE}(Q)$, on some graph G with probabilities



Task: count the number X of red-blue edge subsets that violate Q

- Split the subsets on some parameter e.g., the number of nodes
  $\rightarrow$ $X = X_1, \ldots, X_k$
- Create unweighted copies of G modified with some gadgets e.g., replace each edge by multiple copies of a path
  $\rightarrow$ Created $G_1, \ldots, G_k$
  $\rightarrow$ Call the oracle for $\mathrm{MC}(\mathrm{Q})$ on each to get $N_1, \ldots, N_k$
- Show that each N_i is a linear function of $X_1, \ldots, X_k$, so:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

→ If the matrix is invertible, then we have succeeded

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

→ If the matrix is invertible, then we have succeeded

We can choose gadgets and parameters to get a Vandermonde matrix, and show invertibility via several arithmetical tricks

Conclusion and open problems

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be $1/2$ (model counting)

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be $1/2$ (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be 1/2 (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open

Conclusion and open problems

- **On UCQs**: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be 1/2 (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  - $\rightarrow$ Open, probably challenging

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be 1/2 (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  - $\rightarrow$ Open, probably challenging
- What about disequalities? negations?
  - $\rightarrow$ Poorly understood, even for UCQs

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be 1/2 (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  $\rightarrow$ Open, probably challenging
- What about disequalities? negations?
  $\rightarrow$ Poorly understood, even for UCQs
- What about tractable cases? ...

Conclusion and open problems

- On UCQs: $\text{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\text{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\text{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be $1/2$ (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  → Open, probably challenging
- What about disequalities? negations?
  → Poorly understood, even for UCQs
- What about tractable cases? … …

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
    - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
    - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be 1/2 (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
    - The result on self-join-free CQs extends to that context
    - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  $\rightarrow$ Open, probably challenging
- What about disequalities? negations?
  $\rightarrow$ Poorly understood, even for UCQs
- What about tractable cases? … … … ?

Conclusion and open problems

- On UCQs: $\mathrm{PQE}(Q)$ PTIME for safe UCQs and #P-hard otherwise
- We have shown:
  - $\mathrm{PQE}(Q)$ is #P-hard for any Q closed under homomorphisms unless it is equivalent to a safe UCQ
  - $\mathrm{PQE}(Q)$ is #P-hard for any self-join-free CQ Q even when all probabilities must be $1/2$ (model counting)

Open problems:

- What about higher-arity databases? (hypergraphs)
  - The result on self-join-free CQs extends to that context
  - For queries closed under homomorphisms: still open
- What about unweighted PQE for UCQs or beyond?
  → Open, probably challenging
- What about disequalities? negations?
  → Poorly understood, even for UCQs
- What about tractable cases? … … … ?

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*.

Amarilli, A. and Ceylan, I. I. (2019).
A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs.
Preprint: `https://arxiv.org/abs/1910.02048`.

References ii

Amarilli, A. and Kimelfeld, B. (2019).
Model Counting for Conjunctive Queries Without Self-Joins.
Preprint: https://arxiv.org/abs/1908.07093.

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