Une dichotomie sur l'évaluation de requêtes closes sous homomorphismes sur les graphes probabilistes

Antoine Amarilli[1] and İsmail İlkan Ceylan[2]

29 octobre 2020

[1]Télécom Paris

[2]University of Oxford

Uncertain data management

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

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

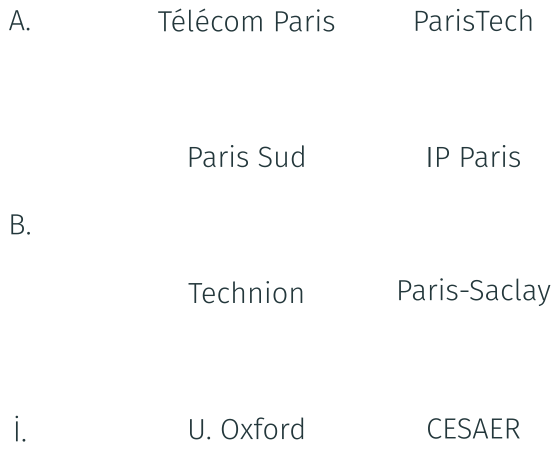
MemberOf	
Télécom Paris	ParisTech
Télécom Paris	IP Paris
Paris Sud	IP Paris
Paris Sud	Paris-Saclay
Technion	CESAER

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

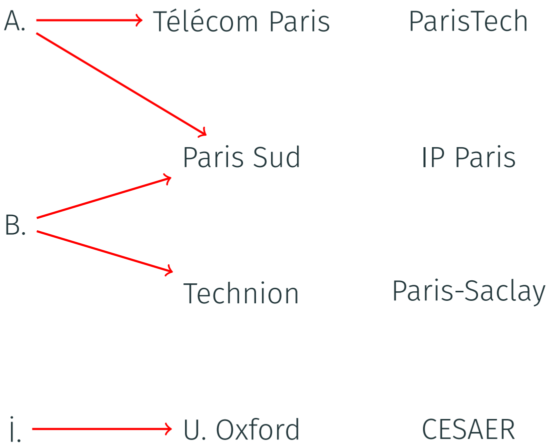


Uncertain data management

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

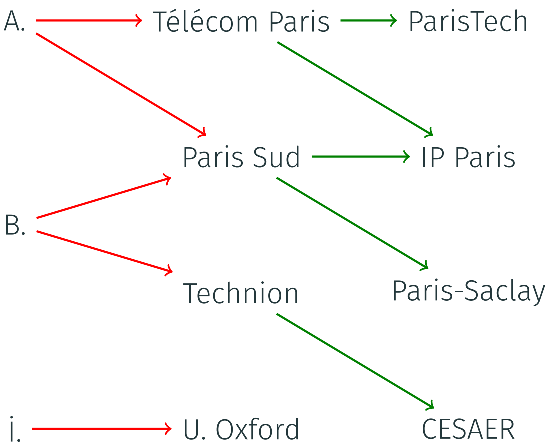


Uncertain data management

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

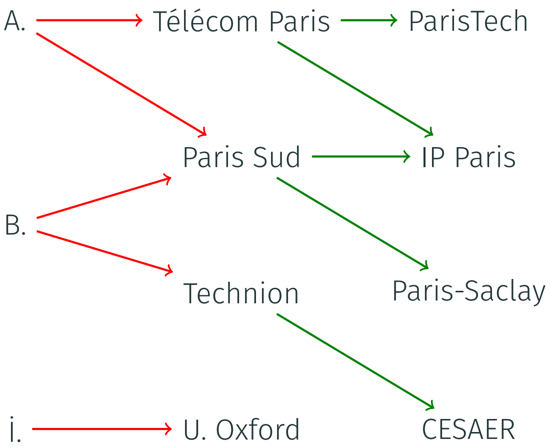


Uncertain data management

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

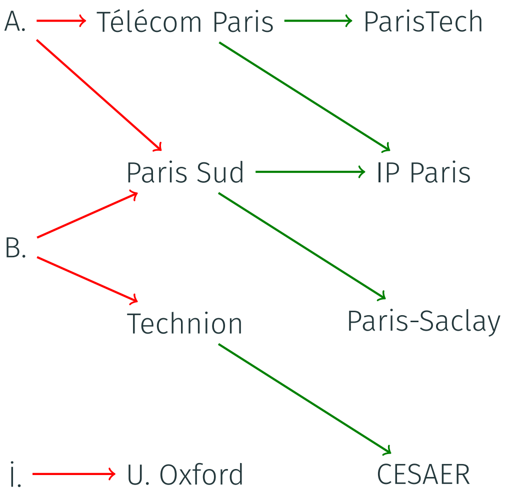
WorksAt	
Antoine	Télécom Paris
Antoine	Paris Sud
Benny	Paris Sud
Benny	Technion
İsmail	U. Oxford

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



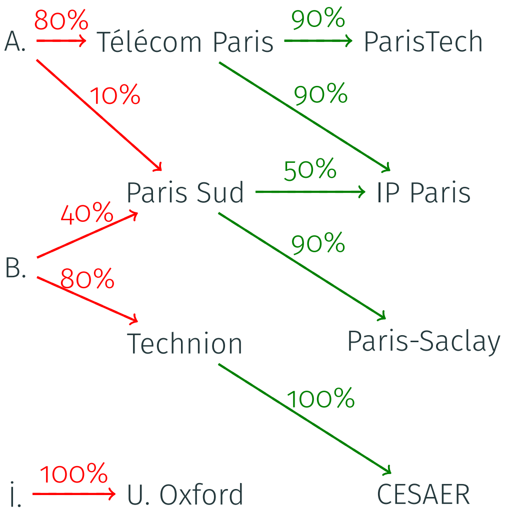
→ **Problem:** we are not **certain** about the true state of the data

Uncertain data model



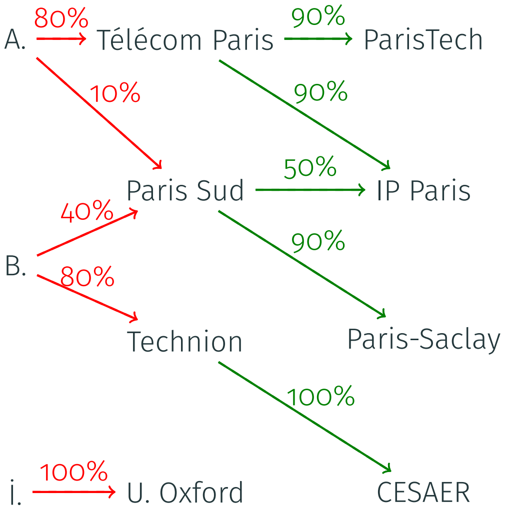
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**

Uncertain data model



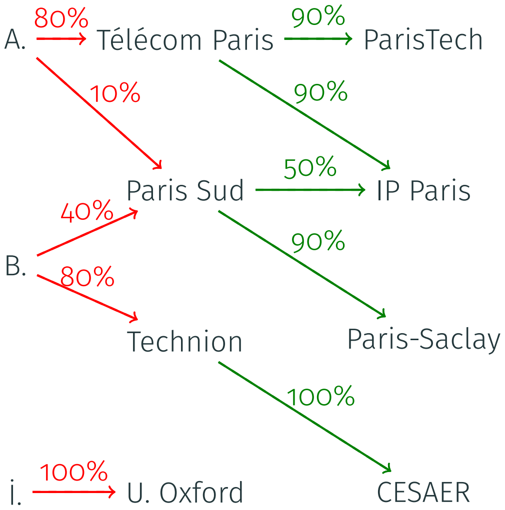
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**

Uncertain data model



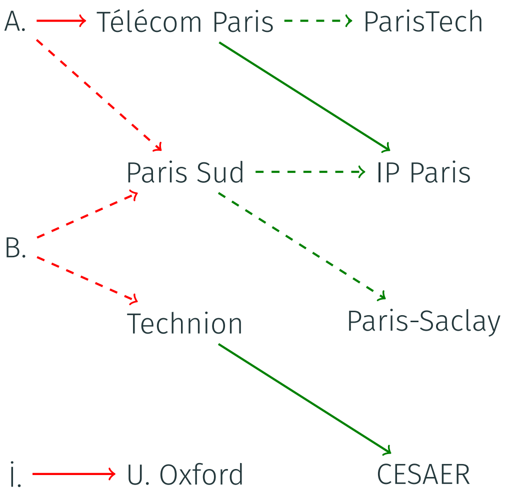
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**

Uncertain data model



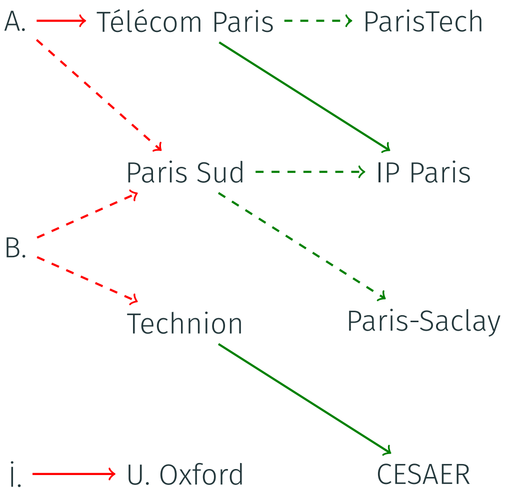
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W:** subset of facts

Uncertain data model



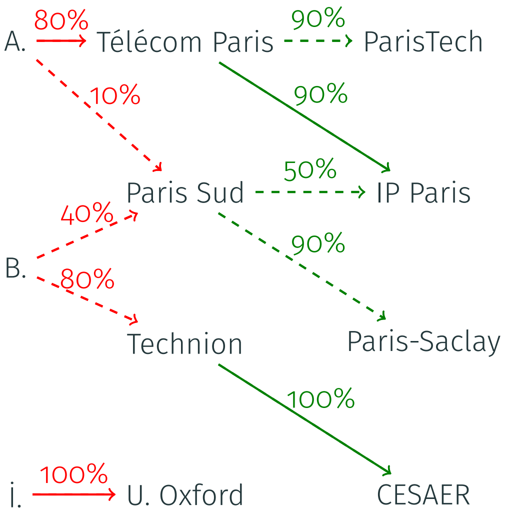
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W:** subset of facts

Uncertain data model



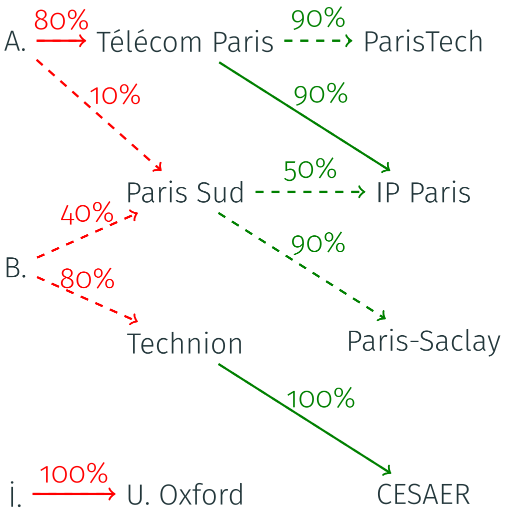
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W:** subset of facts
- What is the **probability** of this possible world?

Uncertain data model



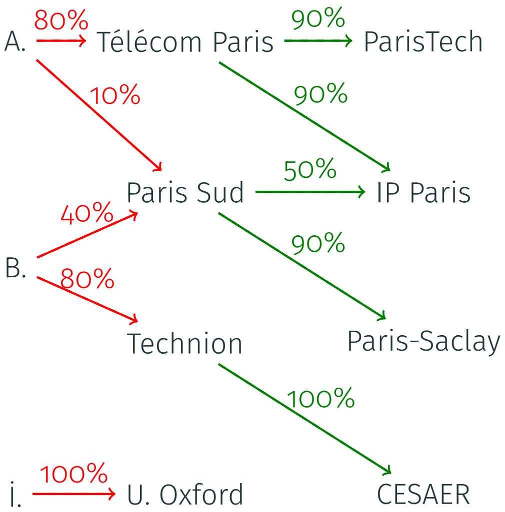
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world *W*:** subset of facts
- What is the **probability** of this possible world?

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%**

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

Queries

- **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., $x \longrightarrow y \longrightarrow z$

Queries

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

Queries

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

Queries

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

$\rightarrow$ **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., $x \longrightarrow y \longrightarrow z$
  - $\rightarrow$ 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**?

$\rightarrow$ **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., $x \longrightarrow y \longrightarrow z$
  - $\rightarrow$ 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**?

$\rightarrow$ **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., $x \longrightarrow y \longrightarrow z$
  - $\rightarrow$ 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**?

$\rightarrow$ **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., $x \longrightarrow y \longrightarrow z$
  $\rightarrow$ 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**?

$\rightarrow$ **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**:
  $\rightarrow$ 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., $x \longrightarrow y \longrightarrow z$
  - $\rightarrow$ 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**?

$\rightarrow$ **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:

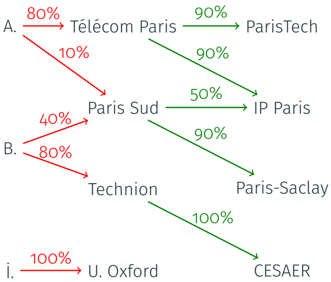
- Generalize **CQs** and **UCQs**, but also **regular path queries** (RPQs), **Datalog**, etc.
- Do not allow for **inequalities** or **negation**
- A homomorphism-closed query can be seen as an **infinite union of CQs**:
  - $\rightarrow$ The query is **bounded** if the union is finite (it is a UCQ), **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: $x \longrightarrow y \longrightarrow z$

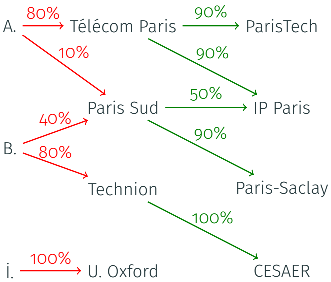
Problem statement: Probabilistic query evaluation (PQE)

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



Problem statement: Probabilistic query evaluation (PQE)

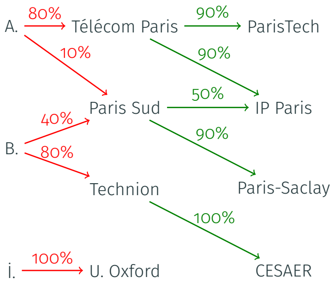
- We **fix** a query Q, for instance the CQ: $x \longrightarrow y \longrightarrow z$
- The **input** is a TID D:



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

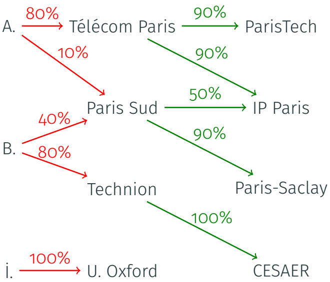


- The **output** is the **total probability** of the worlds which satisfy the query:
  - Formally: $\sum_{W \subseteq D, W \models Q} \Pr(W)$

$\rightarrow$ **Intuition:** the **probability** that the query is true

Problem statement: Probabilistic query evaluation (PQE)

- We **fix** a query Q, for instance the CQ: $x \xrightarrow{\quad} y \xrightarrow{\quad} z$
- 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 $\mathrm{PQE}(Q)$, 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* $\mathrm{PQE}(Q)$ *is in PTIME*
- *All others are unsafe and* $\mathrm{PQE}(Q)$ *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 $\mathrm{PQE}(Q)$ is in **PTIME***
- *All others are **unsafe** and $\mathrm{PQE}(Q)$ is **#P-hard***

- The CQ $x \longrightarrow y \longrightarrow z$ is **safe**, but the CQ $x \longrightarrow y \longrightarrow z \longrightarrow w$ 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* $\mathrm{PQE}(Q)$ *is in* ***PTIME***
- *All others are* ***unsafe*** *and* $\mathrm{PQE}(Q)$ *is* ***#P-hard***

- The CQ $x \longrightarrow y \longrightarrow z$ is **safe**, but the CQ $x \longrightarrow y \longrightarrow z \longrightarrow w$ 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 $\mathrm{PQE}(Q)$ is in **PTIME***
- *All others are **unsafe** and $\mathrm{PQE}(Q)$ is **#P-hard***

- The CQ $x \longrightarrow y \longrightarrow z$ is **safe**, but the CQ $x \longrightarrow y \longrightarrow z \longrightarrow w$ 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 $\mathrm{PQE}(Q)$ is in **PTIME***
- *All others are **unsafe** and $\mathrm{PQE}(Q)$ is **#P-hard***

- The CQ $x \longrightarrow y \longrightarrow z$ is **safe**, but the CQ $x \longrightarrow y \longrightarrow z \longrightarrow w$ 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* $\mathrm{PQE}(Q)$ *is in* ***PTIME***
- *All others are* ***unsafe*** *and* $\mathrm{PQE}(Q)$ *is* ***#P-hard***

- The CQ $x \longrightarrow y \longrightarrow z$ is **safe**, but the CQ $x \longrightarrow y \longrightarrow z \longrightarrow w$ 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 $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ is #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 $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ 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* $\mathrm{PQE}(Q)$ *is in **PTIME***
- *In all other cases,* $\mathrm{PQE}(Q)$ *is **#P-hard***

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the **RPQ** Q: $\textcolor{red}{\longrightarrow}(\textcolor{green}{\longrightarrow})^{*}\textcolor{blue}{\longrightarrow}$

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 $\mathrm{PQE}(Q)$ is in **PTIME***
- *In all other cases, $\mathrm{PQE}(Q)$ is **#P-hard***

- This extends the result of [Jung and Lutz, 2012] and covers RPQs, Datalog, etc.
- Example: the **RPQ** Q: $\longrightarrow (\longrightarrow)^* \longrightarrow$
  - It is **not equivalent to a UCQ**: infinite disjunction $\longrightarrow (\longrightarrow)^i \longrightarrow$ 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 $\mathrm{PQE}(Q)$ is in **PTIME***
- *In all other cases, $\mathrm{PQE}(Q)$ is **#P-hard***

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

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

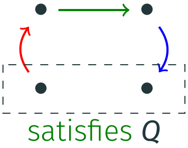
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

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

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



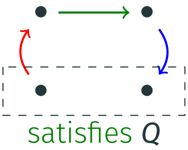
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

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

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



but

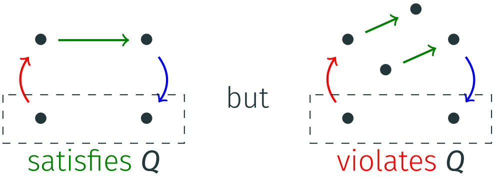
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

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

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



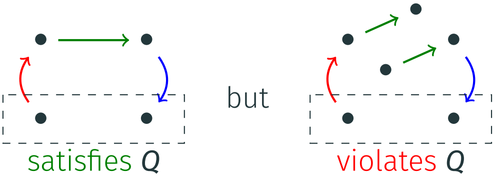
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

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

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

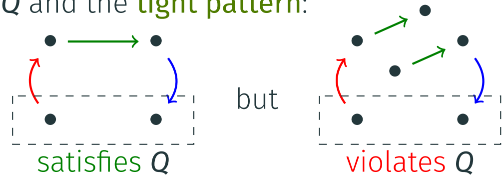


Theorem

Any unbounded query closed under homomorphisms has a tight pattern

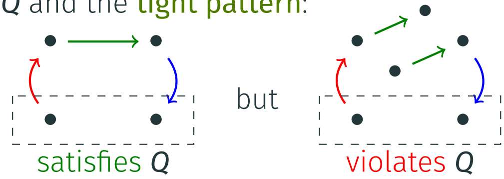
Using tight patterns to show hardness of PQE

- Fix the query Q and the tight pattern:



Using tight patterns to show hardness of PQE

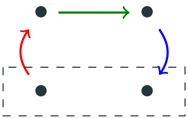
- Fix the query Q and the tight pattern:



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

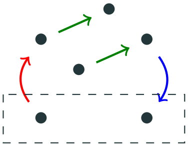
Using tight patterns to show hardness of PQE

- Fix the query Q and the tight pattern:



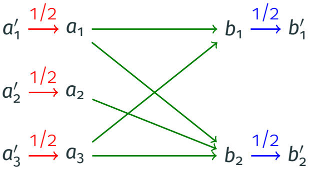
satisfies Q

but

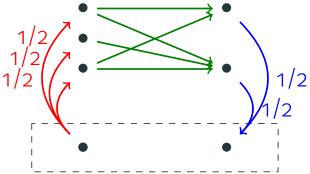


violates Q

- We reduce from PQE for the **unsafe** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$

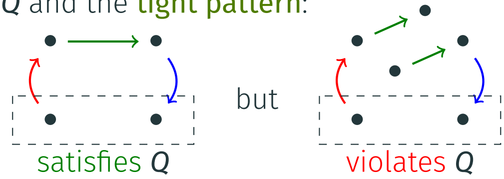


is coded as

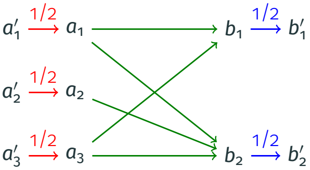


Using tight patterns to show hardness of PQE

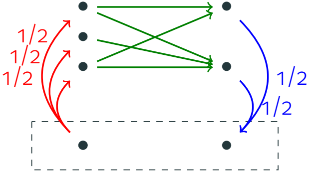
- Fix the query Q and the tight pattern:



- We reduce from PQE for the **unsafe** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



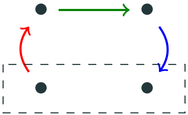
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

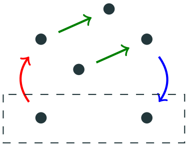
Using tight patterns to show hardness of PQE

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



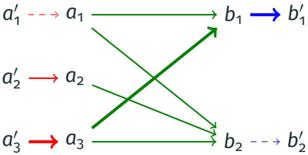
satisfies Q

but

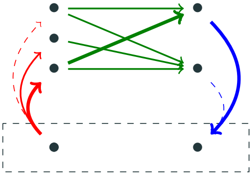


violates Q

- We reduce from PQE for the **unsafe** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



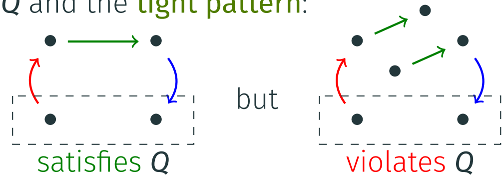
is coded as



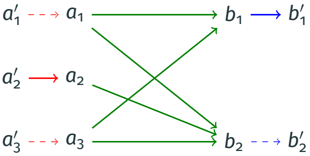
Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

Using tight patterns to show hardness of PQE

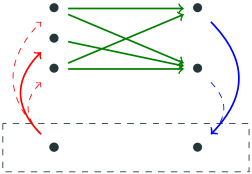
- Fix the query Q and the tight pattern:



- We reduce from PQE for the **unsafe** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



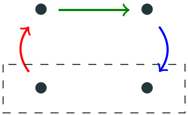
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

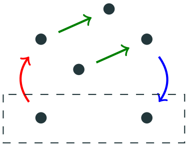
Using tight patterns to show hardness of PQE

- Fix the query Q and the tight pattern:



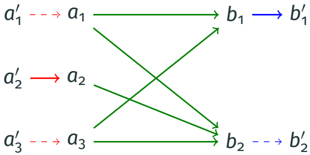
satisfies Q

but

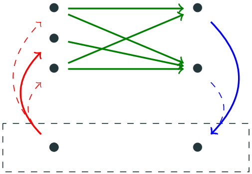


violates Q

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



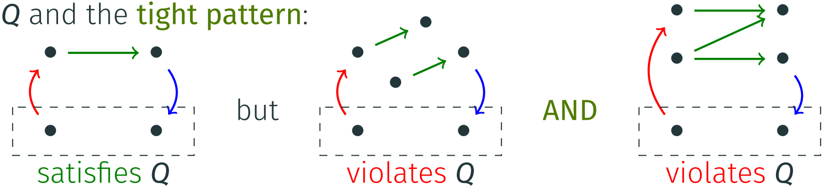
is coded as



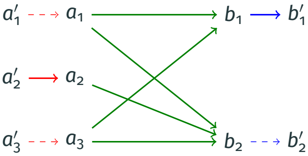
Idea: possible worlds at the left have a path that matches Q_0
iff the corresponding possible world of the TID at the right satisfies the query Q...
... except we need more from the tight pattern!

Using tight patterns to show hardness of PQE

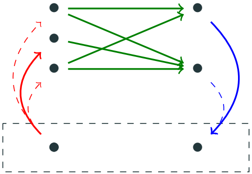
- Fix the query Q and the **tight pattern**:



- We reduce from PQE for the **unsafe** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



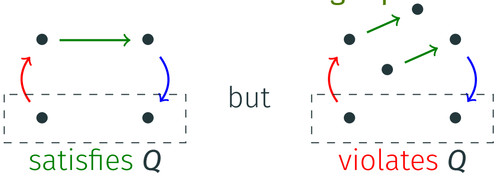
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...
... except we need **more** from the tight pattern!

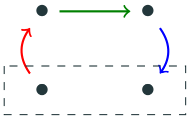
Saving the proof

We know that we have a tight pattern:

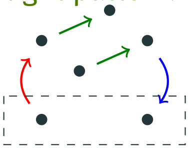


Saving the proof

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



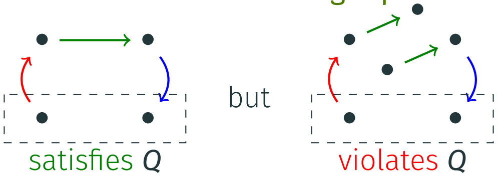
but



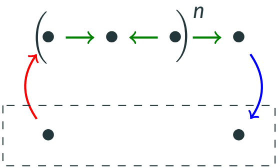
Consider its **iterates**

Saving the proof

We know that we have a tight pattern:

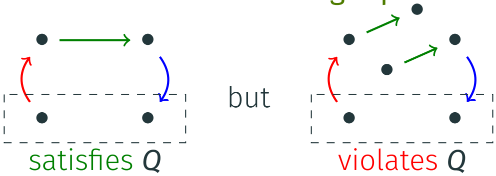


Consider its iterates for each $n \in \mathbb{N}$:

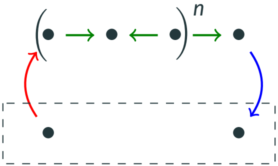


Saving the proof

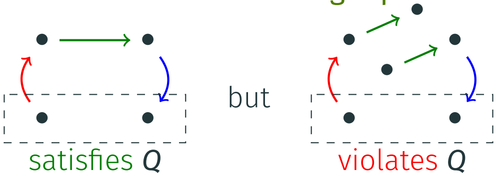
We know that we have a **tight pattern**:



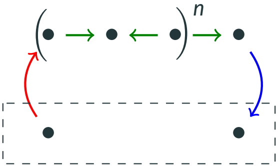
Consider its **iterates** for each $n \in \mathbb{N}$:



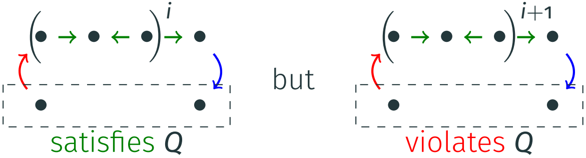
We know that we have a **tight pattern**:



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

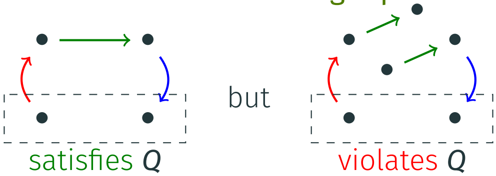


Case 1: some iterate **violates** the query:

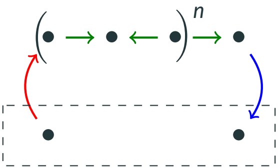


Saving the proof

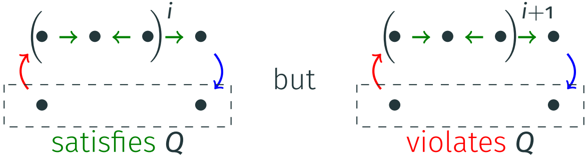
We know that we have a **tight pattern**:



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

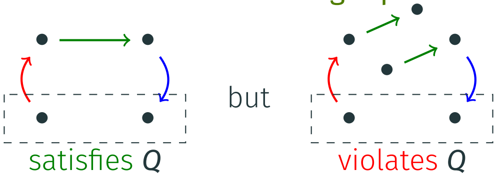


Case 1: some iterate **violates** the query:

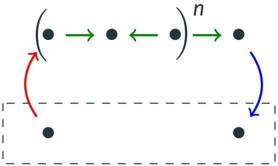


$\rightarrow$ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

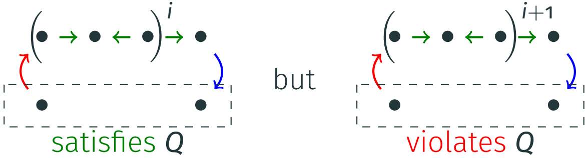
We know that we have a **tight pattern**:



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

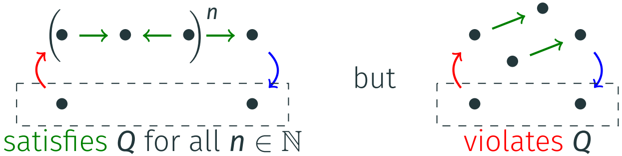


Case 1: some iterate **violates** the query:

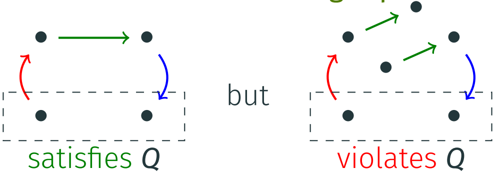


$\rightarrow$ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

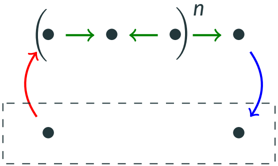
Case 2: all iterates **satisfy** the query:



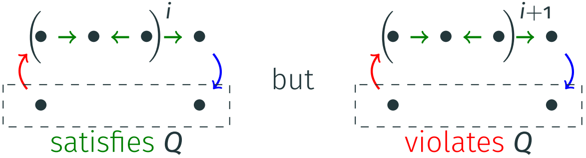
We know that we have a **tight pattern**:



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

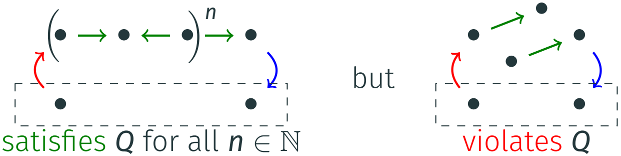


Case 1: some iterate **violates** the query:



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

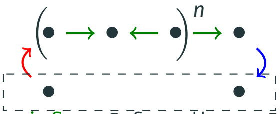
Case 2: all iterates **satisfy** the query:



$\rightarrow$ Call this an **iterable pattern**

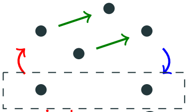
Using iterable patterns to show hardness of PQE

We have an iterable pattern:



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

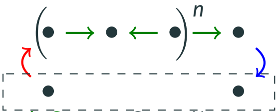
but



violates Q

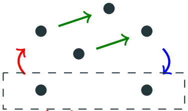
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



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

but



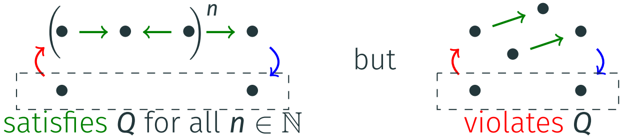
violates Q

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**?

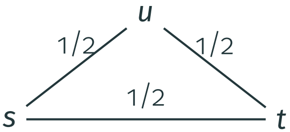
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:

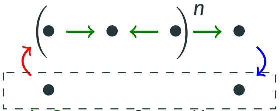


Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- 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**?

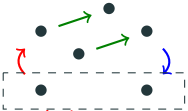


We have an **iterable pattern**:



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

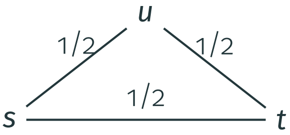
but



violates Q

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

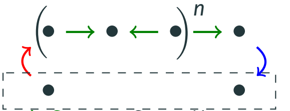
- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as

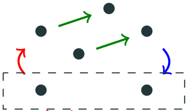
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



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

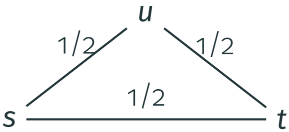
but



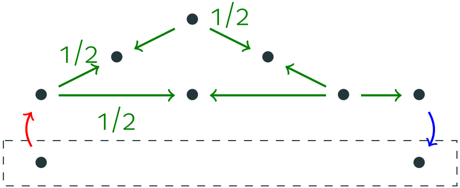
violates Q

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**?



is coded as

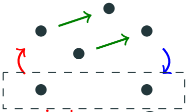


Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:

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

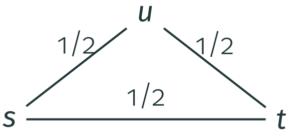
but



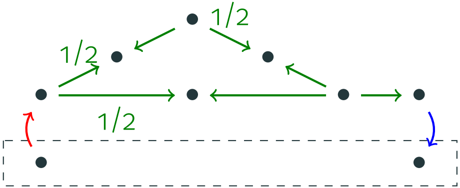
violates Q

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**?



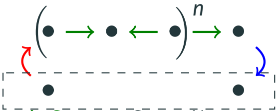
is coded as



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

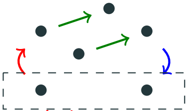
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



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

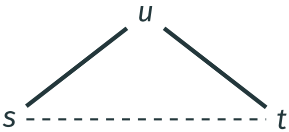
but



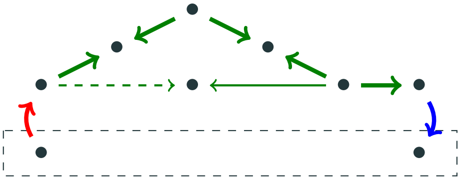
violates Q

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**?



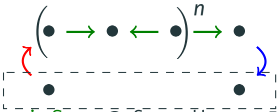
is coded as



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

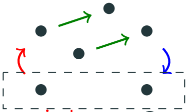
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



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

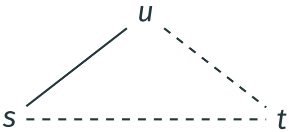
but



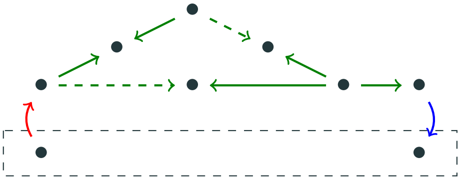
violates Q

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**?



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: $\mathrm{PQE}(Q)$ 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)

Conclusion and open problems

- Our result: $\mathrm{PQE}(Q)$ 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)
- Open problems:
  - The result only applies to graphs, not higher-arity databases

Conclusion and open problems

- Our result: $\mathrm{PQE}(Q)$ 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)

- **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

Conclusion and open problems

- Our result: $\mathrm{PQE}(Q)$ 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)

- **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: $\mathrm{PQE}(Q)$ 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)

- **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 $1/2$?
    - PQE for **non-hierarchical self-join-free CQs** was recently shown to be **#P-hard** in this sense [Amarilli and Kimelfeld, 2020]
    - Similar techniques **may adapt** for our work, but not to the unsafe UCQs...

Conclusion and open problems

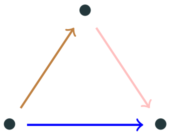
- Our result: $\mathrm{PQE}(Q)$ is **#P-hard** for any query Q **closed under homomorphisms** unless it is equivalent to a safe UCQ
  - $\rightarrow$ Dichotomy for probabilistic query evaluation over **homomorphism-closed** queries
  - $\rightarrow$ Implies intractability for RPQs, Datalog queries, ontology-mediated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

- **Open problems:**
  - The result only applies to **graphs**, not higher-arity databases
    - We **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 $1/2$?
    - PQE for **non-hierarchical self-join-free CQs** was recently shown to be **#P-hard** in this sense [Amarilli and Kimelfeld, 2020]
    - Similar techniques **may adapt** for our work, but not to the unsafe UCQs...

Thanks for your attention!

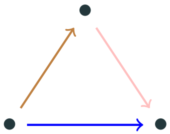
Why can we always find tight patterns?

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

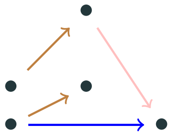


Why can we always find tight patterns?

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

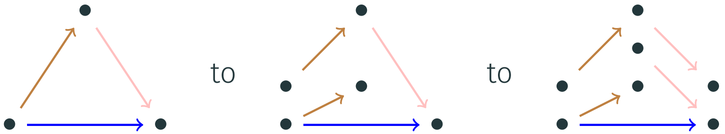


to



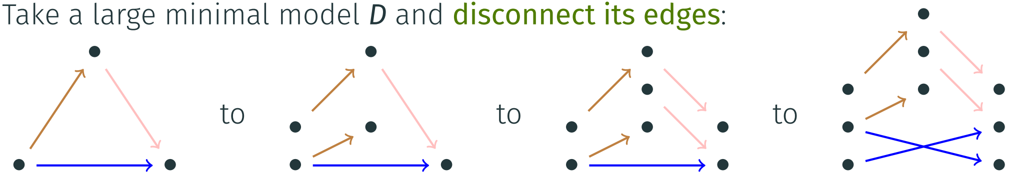
Why can we always find tight patterns?

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



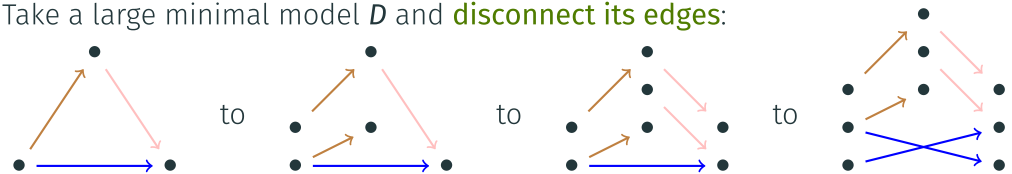
Why can we always find tight patterns?

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



Why can we always find tight patterns?

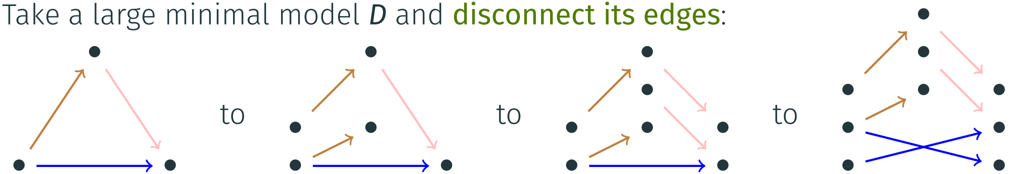
- Unbounded queries have arbitrarily large minimal models
- Take a large minimal model D and disconnect its edges:



- If Q becomes false at one step, then we have found a tight pattern

Why can we always find tight patterns?

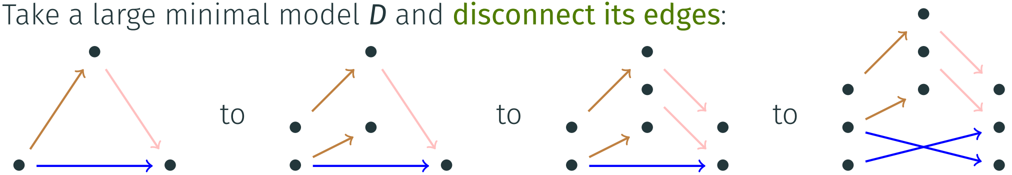
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**

Why can we always find tight patterns?

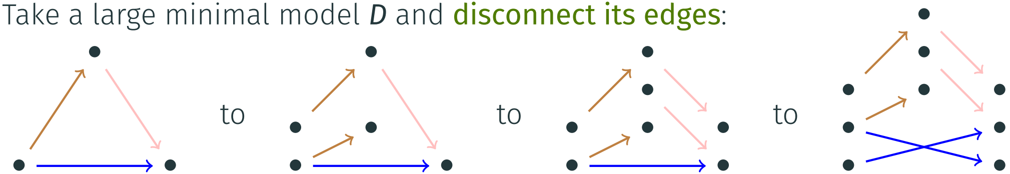
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- 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'

Why can we always find tight patterns?

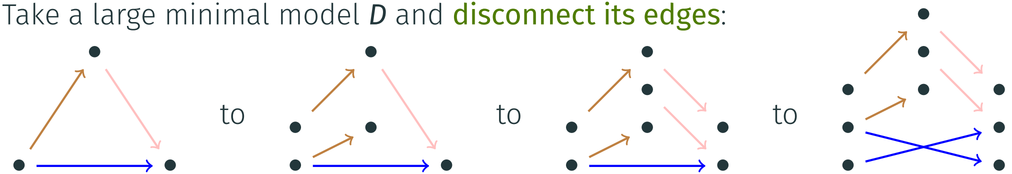
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- 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

Why can we always find tight patterns?

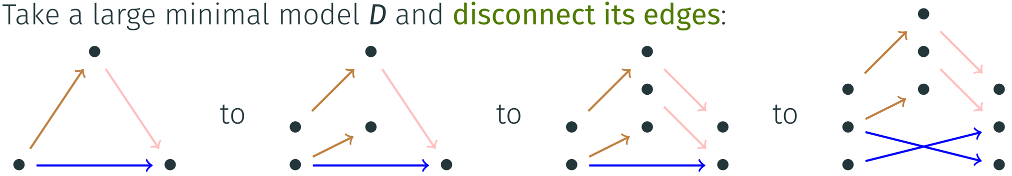
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- 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

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

How to show #P-hardness for PQE

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

- 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

How to show #P-hardness for PQE

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

- 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 : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- 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 $Y_1, \ldots, Y_m$

How to show #P-hardness for PQE

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

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \vee y) \wedge z$, 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 $X_i \wedge Y_j$

How to show #P-hardness for PQE

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

- 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 : $x \longrightarrow y \longrightarrow z \longrightarrow w$

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

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

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

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

How to show #P-hardness for PQE

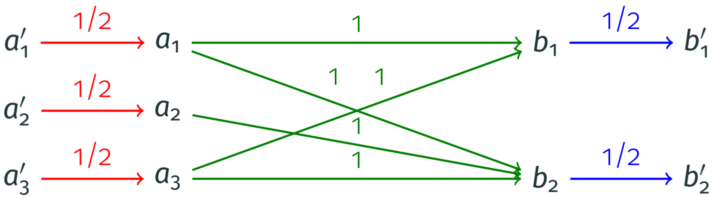
How to show the **#P-hardness** of PQE for the **unsafe** query $Q: x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \vee y) \wedge z$, 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 $X_i \wedge Y_j$
- Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

How to show #P-hardness for PQE

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

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

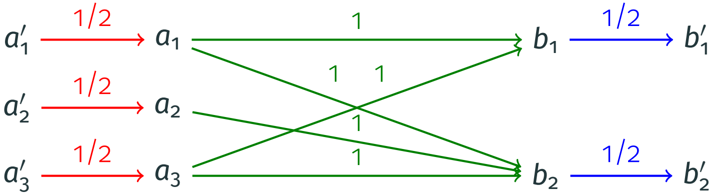


How to show #P-hardness for PQE

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

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \vee y) \wedge z$, 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 $X_i \wedge Y_j$

- Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$



Idea: Satisfying valuations of ϕ correspond to **possible worlds** with a **match** of Q

References i

Amarilli, A. and Kimelfeld, B. (2020).
Uniform Reliability of Self-Join-Free Conjunctive Queries.
Under review.

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

Fink, R. and Olteanu, D. (2016).
Dichotomies for queries with negation in probabilistic databases.
ACM Transactions on Database Systems, 41(1):4:1–4:47.

Jung, J. C. and Lutz, C. (2012).
Ontology-based access to probabilistic data with OWL QL.
In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, pages 182–197.