A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs

Antoine Amarilli

October 29, 2020

Télécom Paris

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Uncertain data management

Relational databases manage data, represented here as a labeled graph

Uncertain data management

Relational databases manage **data**, represented here as a **labeled graph**

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

Uncertain data management

Relational databases manage data, represented here as a labeled graph

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

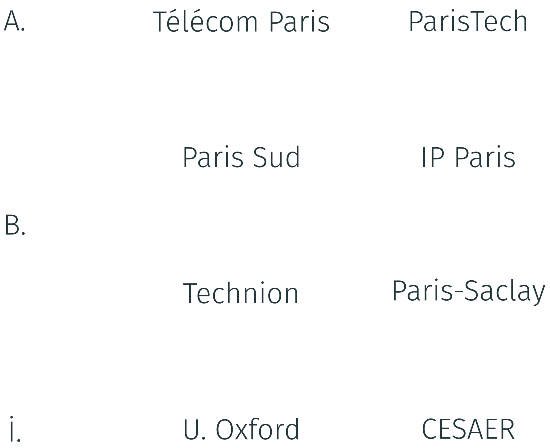
MemberOf	
Télécom Paris	ParisTech
Télécom Paris	IP Paris
Paris Sud	IP Paris
Paris Sud	Paris-Saclay
Technion	CESAER

Uncertain data management

Relational databases manage **data**, represented here 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

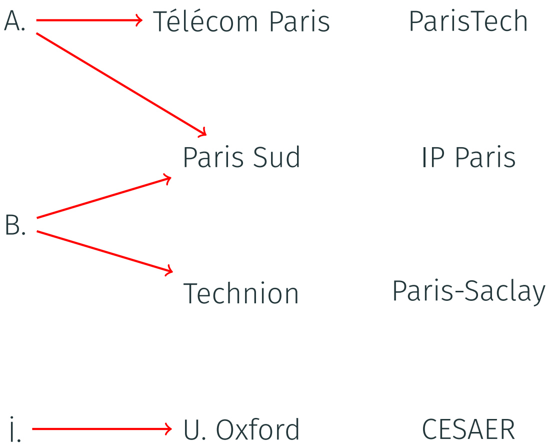
Relational databases manage **data**, represented here as a **labeled graph**

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

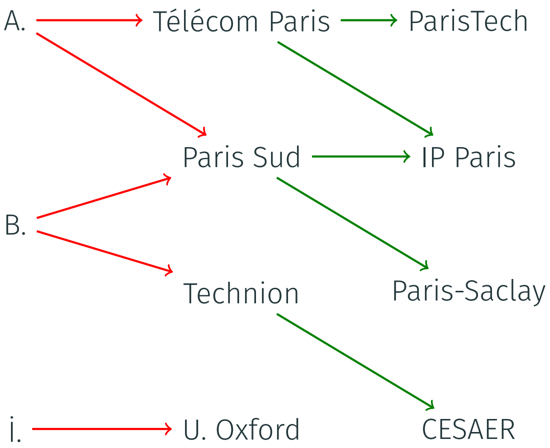


Uncertain data management

Relational databases manage **data**, represented here 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

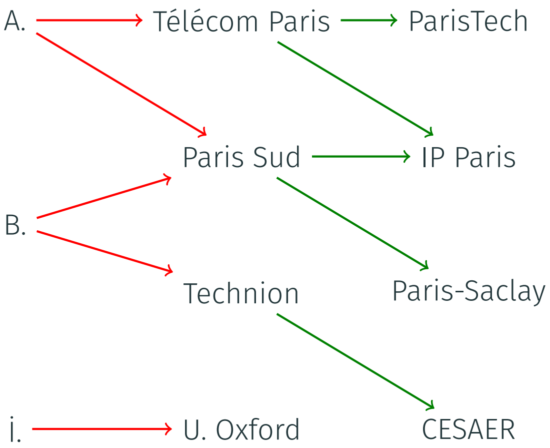


Uncertain data management

Relational databases manage **data**, represented here as a **labeled graph**

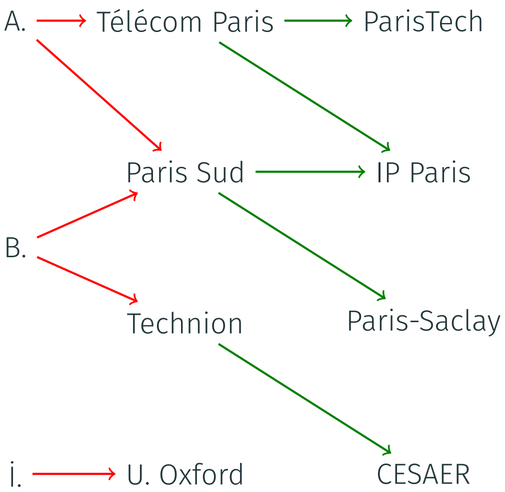
WorksAt	
Antoine	Télécom Paris
Antoine	Paris Sud
Benny	Paris Sud
Benny	Technion
İsmail	U. Oxford

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



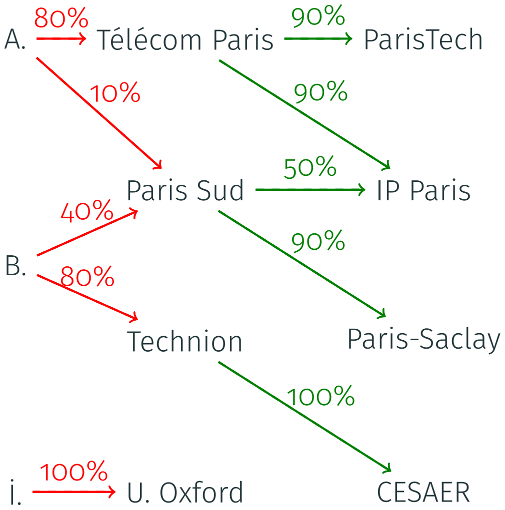
→ **Problem:** we are not **certain** about the true state of the data

Uncertain data model



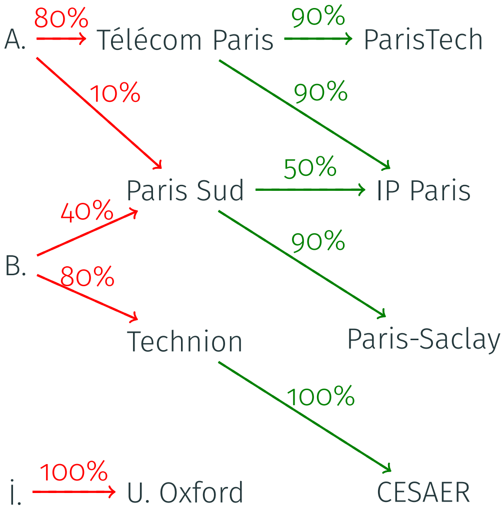
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**

Uncertain data model



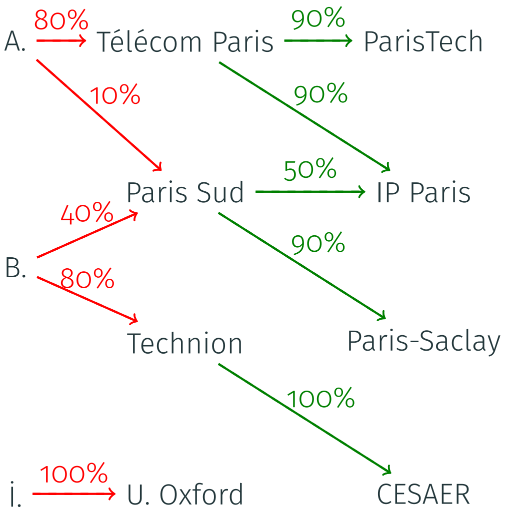
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**

Uncertain data model



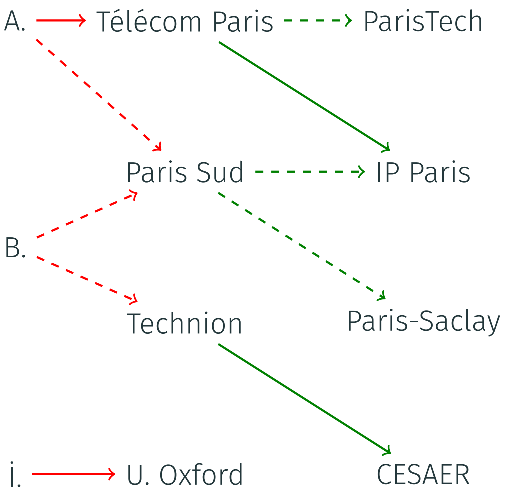
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**

Uncertain data model



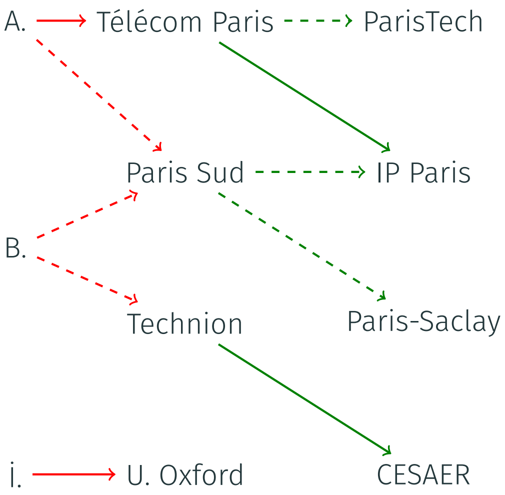
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W**: subset of facts

Uncertain data model



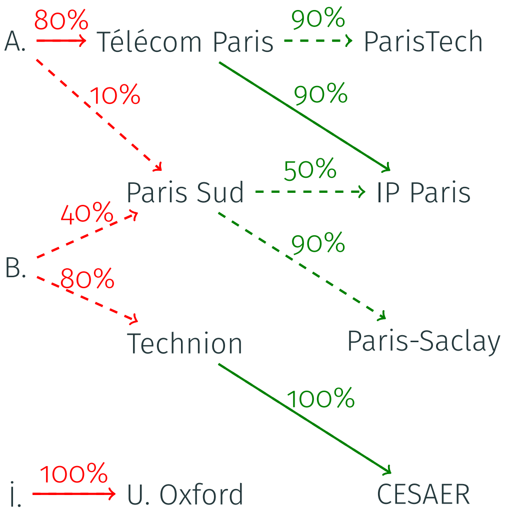
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W:** subset of facts

Uncertain data model



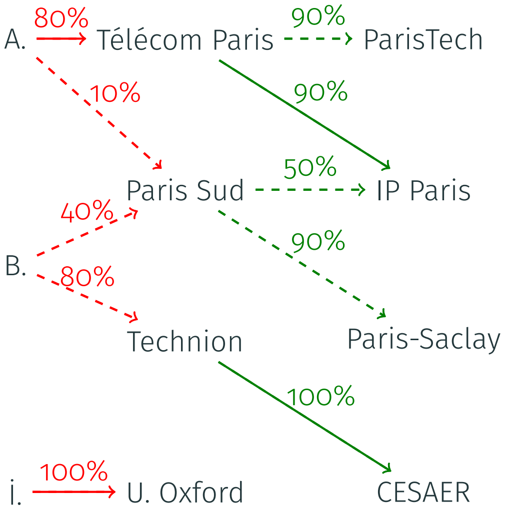
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world W:** subset of facts
- What is the **probability** of this possible world?

Uncertain data model



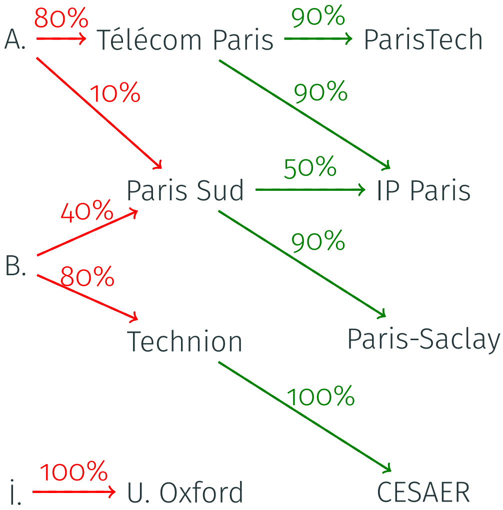
- Uncertain data model: **TID**, for **tuple-independent database**
- Each fact (edge) carries a **probability**
- Each fact exists with its given **probability**
- All facts are **independent**
- **Possible world *W*:** subset of facts
- What is the **probability** of this possible world?

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%**

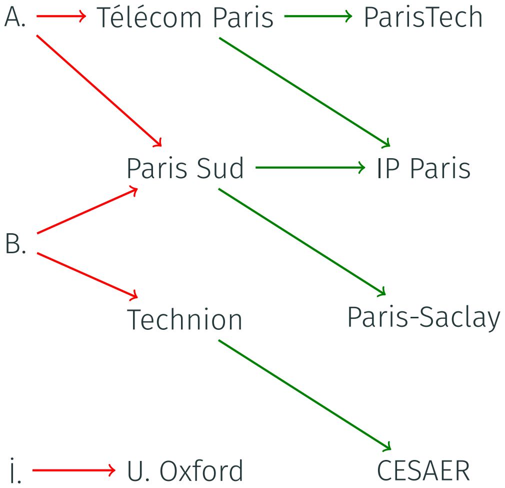
Uncertain data model



- 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%**

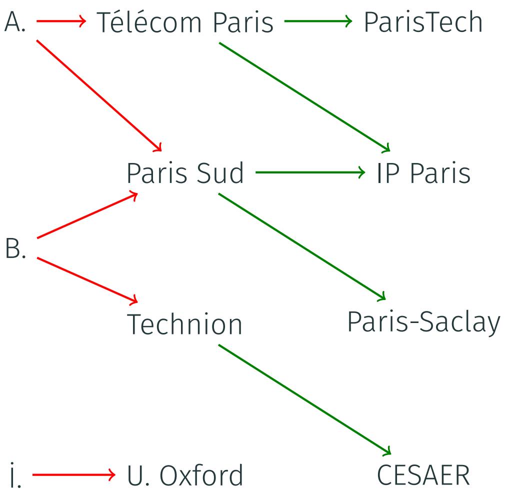
$$\Pr(W) = \left(\prod_{F \in W} \Pr(F)\right) \times \left(\prod_{F \notin W} (1 - \Pr(F))\right)$$

Queries



Central database task: **evaluate queries**

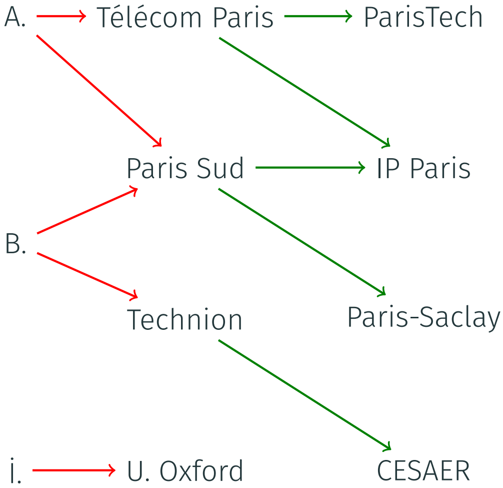
Queries



Central database task: **evaluate queries**

"Is there some person ***x*** *employed in an institution who is part of a consortium* ***z****?"*

Queries

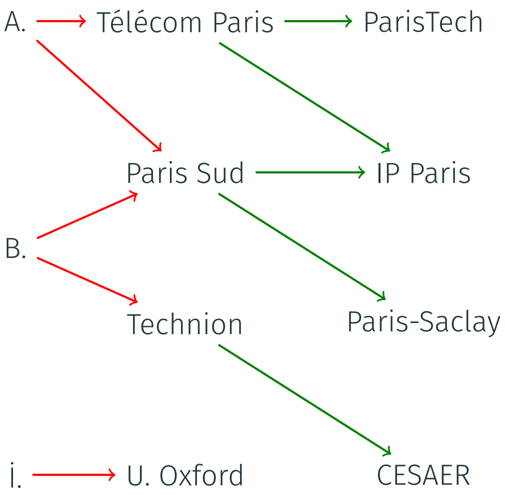


Central database task: **evaluate queries**

"Is there some person ***x*** *employed in an institution who is part of a consortium* ***z****?"*

$$Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$$

Queries



Central database task: **evaluate queries**

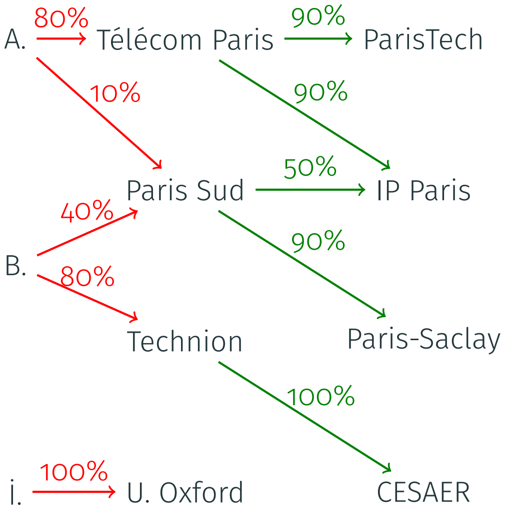
"Is there some person x *employed in an institution who is part of a consortium* z*?"*

$$Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$$

Result on this graph:

x	z
A.	ParisTech
A.	IP Paris
A.	Paris-Saclay
B.	IP Paris
B.	Paris-Saclay
B.	CESAER

Queries



Central database task: **evaluate queries**

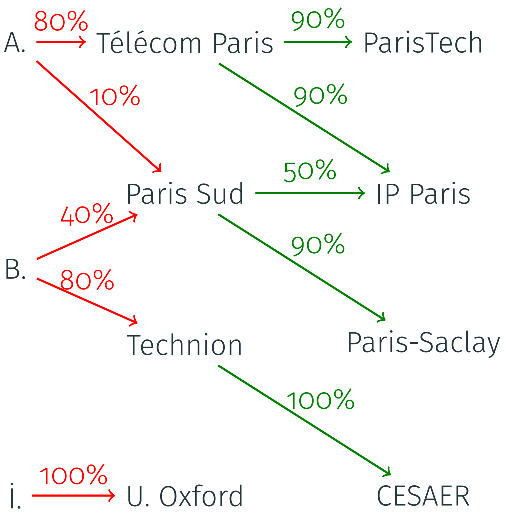
*"Is there some person **x** employed in an institution who is part of a consortium **z**?"*

$$Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$$

Result on this graph:

x	z
A.	ParisTech
A.	IP Paris
A.	Paris-Saclay
B.	IP Paris
B.	Paris-Saclay
B.	CESAER

Queries



Central database task: **evaluate queries**

"Is there some person x *employed in an institution who is part of a consortium* z*?"*

$$Q(x,z) : \exists y \quad x \longrightarrow y \longrightarrow z$$

Result on this graph:

x	z	
A.	ParisTech	72%
A.	IP Paris	99.1%
A.	Paris-Saclay	9%
B.	IP Paris	20%
B.	Paris-Saclay	36%
B.	CESAER	80%

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to YES/NO queries:

- Query: maps a graph to YES/NO

Why can we get away with that?

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to YES/NO queries:

- Query: maps a graph to YES/NO

Why can we get away with that?

- Consider a query: $Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to **YES/NO queries**:

- **Query:** maps a graph to YES/NO

Why can we get away with that?

- Consider a query: $Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$
- Consider **each possible choice** of (x, z), e.g., (A., CESAER)

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to **YES/NO queries**:

- **Query:** maps a graph to YES/NO

Why can we get away with that?

- Consider a query: $Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$
- Consider **each possible choice** of (x, z), e.g., (A., CESAER)
- The query Q(A., CESAER) is a **YES/NO query**:

$$Q(\text{A., CESAER.}) : \exists y \quad x \longrightarrow y \longrightarrow z$$

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to **YES/NO queries**:

- **Query:** maps a graph to YES/NO

Why can we get away with that?

- Consider a query: $Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$
- Consider **each possible choice** of (x, z), e.g., (A., CESAER)
- The query Q(A., CESAER) is a **YES/NO query**:

$$Q(\text{A., CESAER.}) : \exists y \quad x \longrightarrow y \longrightarrow z$$

- The number of choices for (x, z) is **polynomial** in the input graph

Restricting to YES/NO queries

To make the problem simpler to study, we will restrict to **YES/NO queries**:

- **Query:** maps a graph to YES/NO

Why can we get away with that?

- Consider a query: $Q(x, z) : \exists y \quad x \longrightarrow y \longrightarrow z$
- Consider **each possible choice** of (x, z), e.g., (A., CESAER)
- The query Q(A., CESAER) is a **YES/NO query**:

$$Q(\text{A., CESAER.}) : \exists y \quad x \longrightarrow y \longrightarrow z$$

- The number of choices for (x, z) is **polynomial** in the input graph

→ From now on, all queries are **YES/NO queries**,
so we have just one YES/NO answer to compute, or **just one** probability

Query languages

Which kinds of queries do we want to express?

- Conjunctive query (CQ): can I find a match of a pattern?
  - e.g., $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

Query languages

Which kinds of queries do we want to express?

- **Conjunctive query** (CQ): can I find a match of a **pattern**?
  - e.g., $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

$\rightarrow$ We want a **homomorphism** from the pattern to the graph (not necessarily **injective**)

$\rightarrow$ Formally: an **existentially quantified conjunction of atoms (edges)**

Query languages

Which kinds of queries do we want to express?

- Conjunctive query (CQ): can I find a match of a pattern?
  - e.g., $\exists x\,y\,z \quad x \longrightarrow y \longrightarrow z$

  $\rightarrow$ We want a homomorphism from the pattern to the graph (not necessarily injective)

  $\rightarrow$ Formally: an existentially quantified conjunction of atoms (edges)

- Union of conjunctive queries (UCQ): can I find a match of some pattern?
  - e.g., $(\exists x\,y\,z \quad x \longrightarrow y \longrightarrow z \;) \vee (\exists x\,y\,z\,w \quad x \longrightarrow y \quad z \longrightarrow w \;)$

Query languages

Which kinds of queries do we want to express?

- **Conjunctive query** (CQ): can I find a match of a **pattern**?
  - e.g., $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

  → We want a **homomorphism** from the pattern to the graph (not necessarily **injective**)

  → Formally: an **existentially quantified conjunction of atoms (edges)**

- **Union of conjunctive queries** (UCQ): can I find a match of **some pattern**?
  - e.g., $(\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z \quad) \vee (\exists x\, y\, z\, w \quad x \longrightarrow y \quad z \longrightarrow w \quad)$

  → Formally: a **finite disjunction** of CQs

Query languages

Which kinds of queries do we want to express?

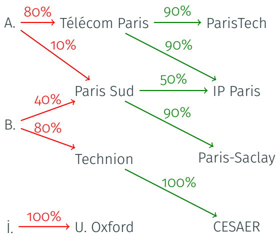
- **Conjunctive query** (CQ): can I find a match of a **pattern**?
  - e.g., $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$
  - $\rightarrow$ We want a **homomorphism** from the pattern to the graph (not necessarily **injective**)
  - $\rightarrow$ Formally: an **existentially quantified conjunction of atoms (edges)**
- **Union of conjunctive queries** (UCQ): can I find a match of **some pattern**?
  - e.g., $(\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z\ ) \vee (\exists x\, y\, z\, w \quad x \longrightarrow y \quad z \longrightarrow w\ )$
  - $\rightarrow$ Formally: a **finite disjunction** of CQs
- **Regular path queries** (RPQ): can I find a match of a **regular path**?
  - e.g., $\exists x\, y \quad x \ \longrightarrow (\longrightarrow)^{*} \longrightarrow y$

Problem statement: Probabilistic query evaluation (PQE)

- We fix a query Q, for instance the CQ: $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

Problem statement: Probabilistic query evaluation (PQE)

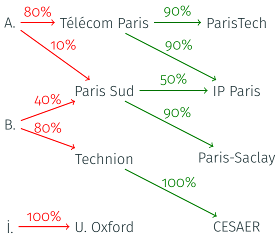
- We **fix** a query Q, for instance the CQ: $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$
- The **input** is a TID D:



Problem statement: Probabilistic query evaluation (PQE)

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

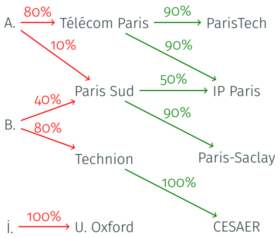
- The **input** is a TID D:



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



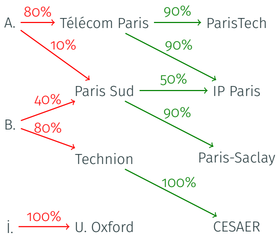
- The **output** is the **total probability** of the worlds which satisfy the query:
  - Formally: $\sum_{W \subseteq D, W \models Q} \Pr(W)$

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

Problem statement: Probabilistic query evaluation (PQE)

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

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

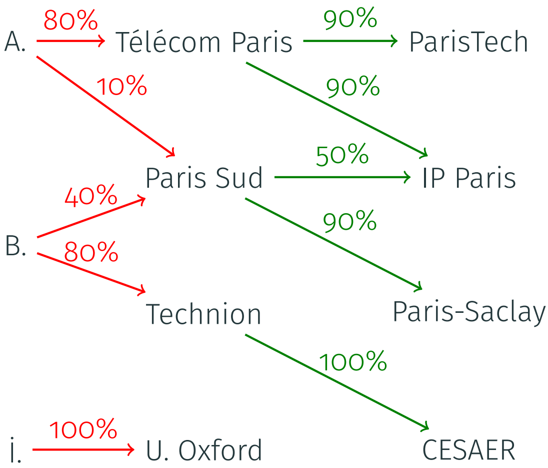


- The **output** is the **total probability** of the worlds which satisfy the query:
  - Formally: $\sum_{W \subseteq D, W \models Q} \Pr(W)$

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

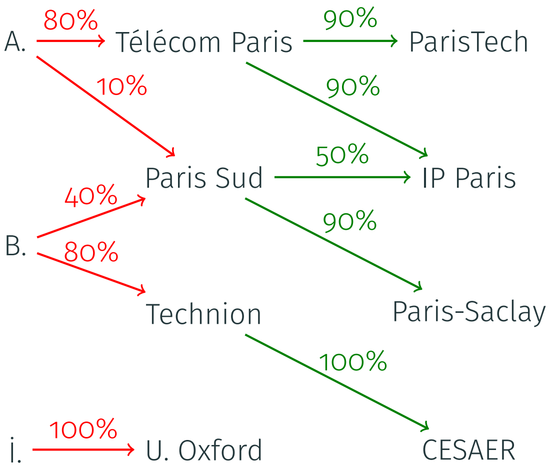
- We can always compute the probability in exponential time (go over all possibilities)

PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

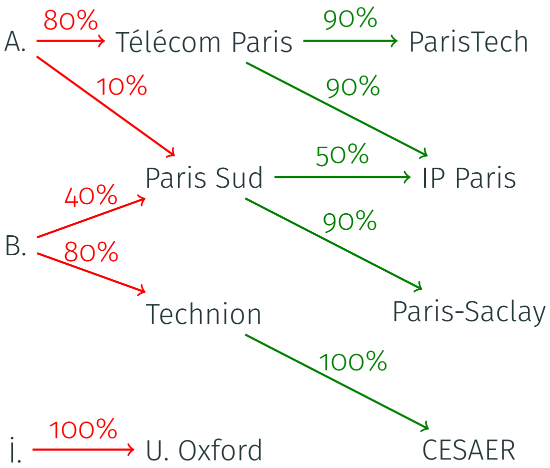
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $1 - x$

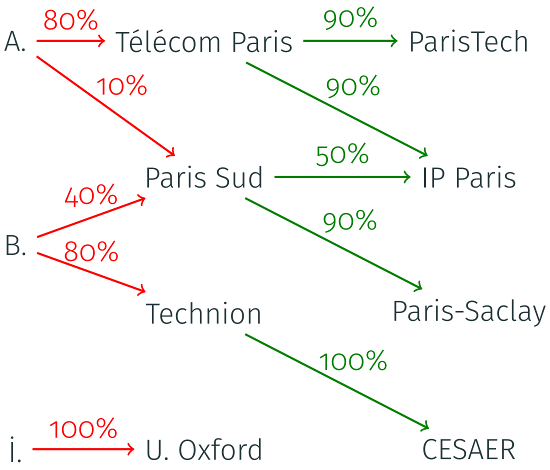
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $\mathbf{1} - x$
- There is no match of the query iff **every red edge is not kept**

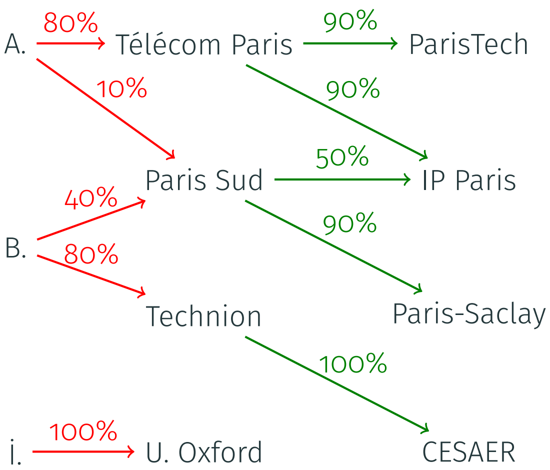
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $\mathbf{1-x}$
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is:

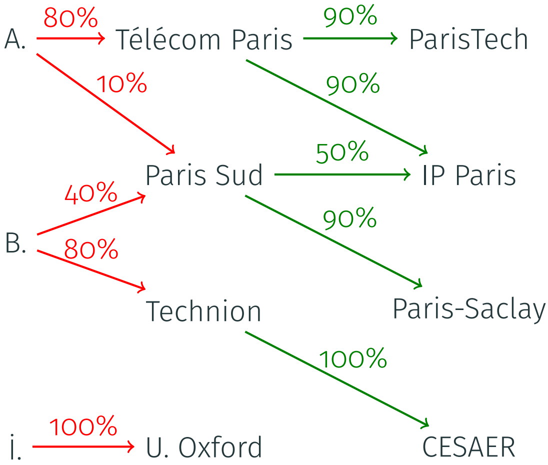
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $1 - x$
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is: $(1 - 80\%)$

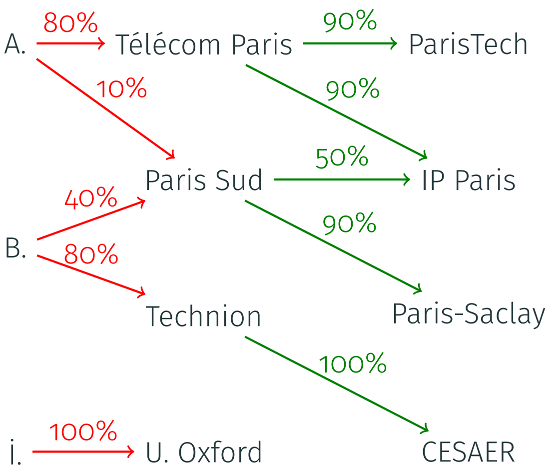
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is **$1 - x$**
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is: $(1 - 80\%) \times (1 - 10\%)$

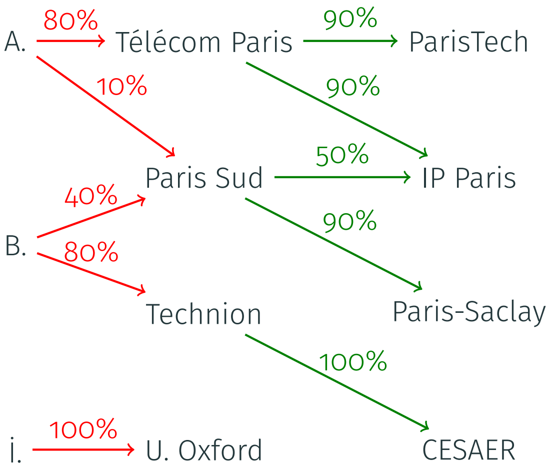
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $\mathbf{1 - x}$
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is: $(1 - 80\%) \times (1 - 10\%) \times (1 - 40\%) \times (1 - 80\%) \times (1 - 100\%)$

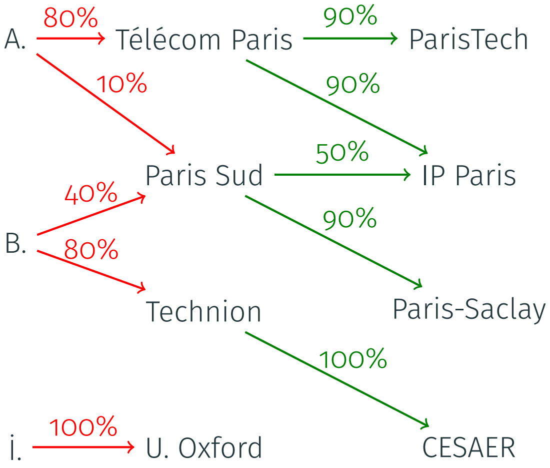
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is **$1 - x$**
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is: $(1 - 80\%) \times (1 - 10\%) \times (1 - 40\%) \times (1 - 80\%) \times (1 - 100\%)$
- This gives **$x = 0\%$**, so the query has probability **100%**

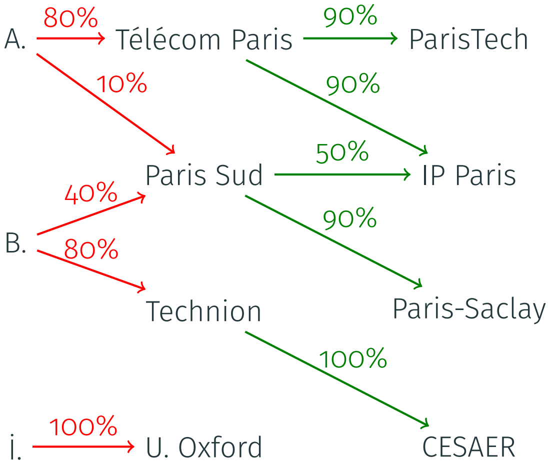
PQE: a simple example



Find the probability of: $\exists x\, y \quad x \longrightarrow y$

- It's easier to compute the probability x that there is **no match of the query**
  - $\rightarrow$ The probability we want is $\mathbf{1 - x}$
- There is no match of the query iff **every red edge is not kept**
- These choices are independent, so x is: $(1 - 80\%) \times (1 - 10\%) \times (1 - 40\%) \times (1 - 80\%) \times (1 - 100\%)$
- This gives $\mathbf{x = 0\%}$, so the query has probability **100%**
- This process is in **polynomial time**

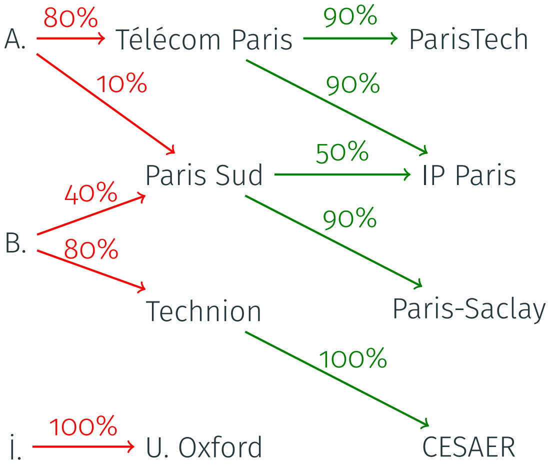
PQE: a more complicated example



How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

PQE: a more complicated example

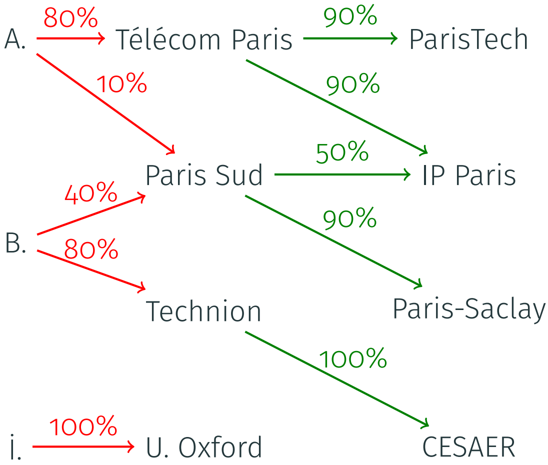


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \xrightarrow{} y \xrightarrow{} z$

- Key insight: consider all possible choices for the middle variable y

PQE: a more complicated example

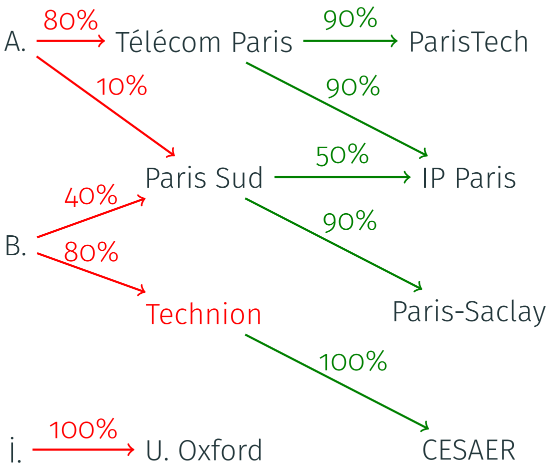


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 -$

PQE: a more complicated example

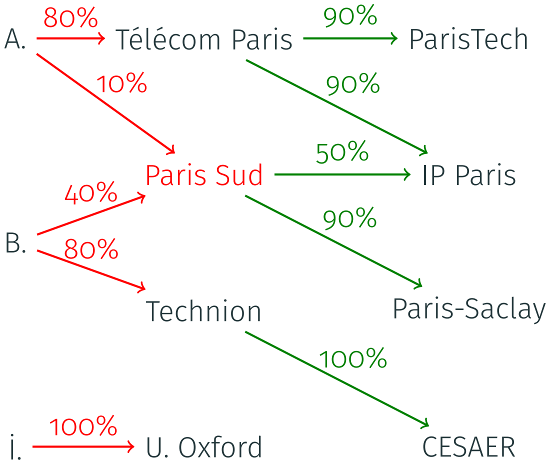


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times$

PQE: a more complicated example

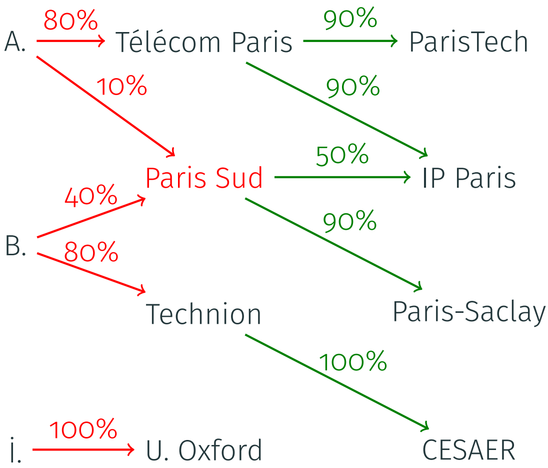


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 -$

PQE: a more complicated example

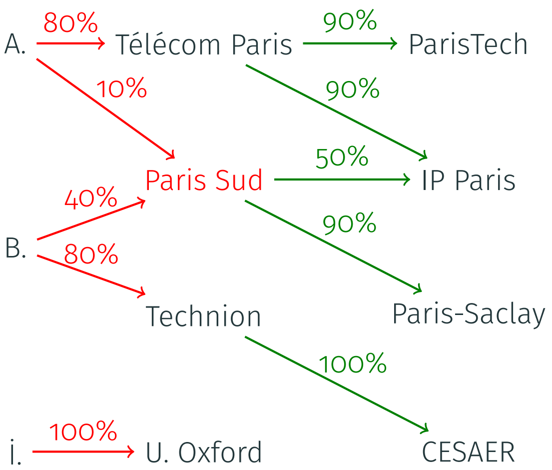


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 -$
  $) \times (1 -$ $)$

PQE: a more complicated example

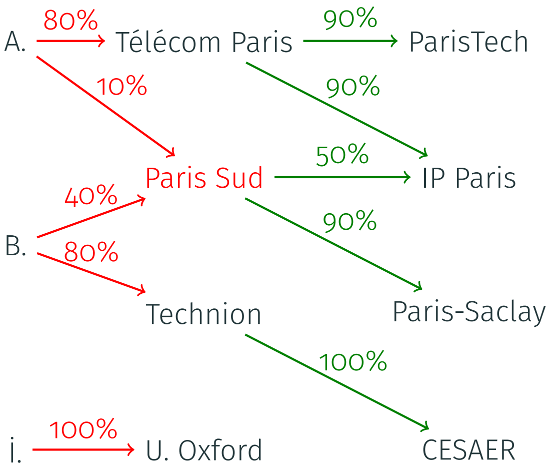


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - \quad)$

PQE: a more complicated example

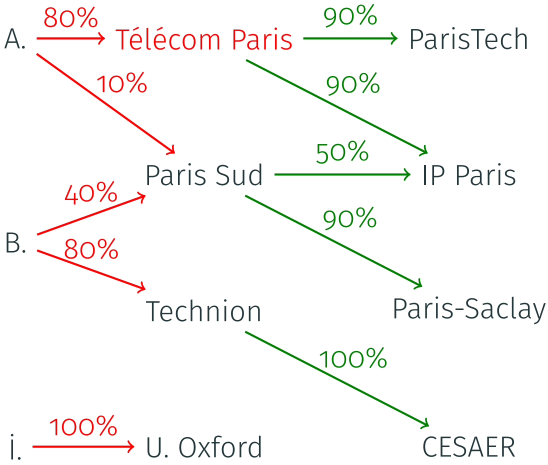


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - (1 - 50\%) \times (1 - 90\%)))$

PQE: a more complicated example

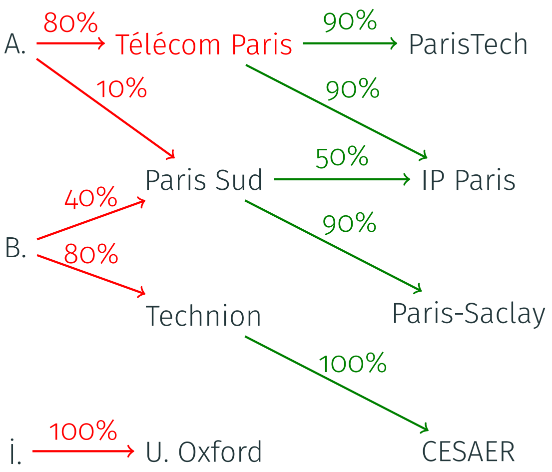


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - (1 - 50\%) \times (1 - 90\%)))$

PQE: a more complicated example

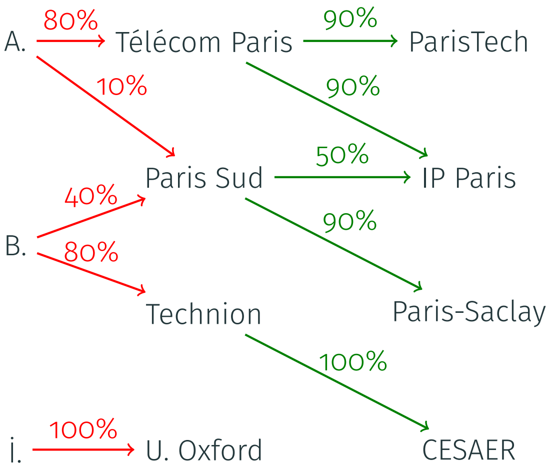


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - (1 - 50\%) \times (1 - 90\%))) \times (1 - 80\% \times (1 - (1 - 90\%) \times (1 - 90\%)))$,

PQE: a more complicated example

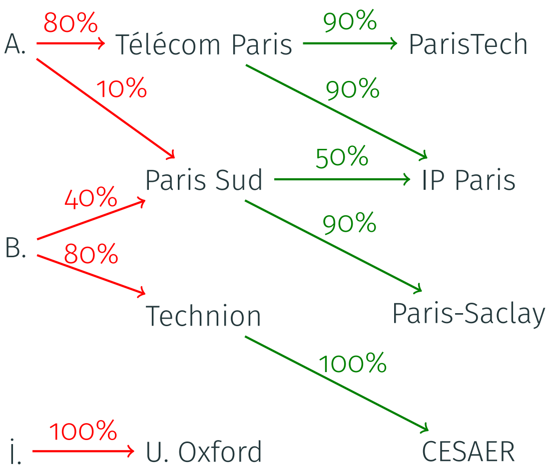


How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - (1 - 50\%) \times (1 - 90\%))) \times (1 - 80\% \times (1 - (1 - 90\%) \times (1 - 90\%)))$, i.e., 97.65792%

PQE: a more complicated example



How to compute the probability of the query from the previous slide?

$\exists x\, y\, z \quad x \xrightarrow{\hspace{2cm}} y \xrightarrow{\hspace{2cm}} z$

- Key insight: consider all possible choices for the middle variable y
- $1 - (1 - 80\%) \times (1 - (1 - (1 - 10\%) \times (1 - 40\%)) \times (1 - (1 - 50\%) \times (1 - 90\%))) \times (1 - 80\% \times (1 - (1 - 90\%) \times (1 - 90\%)))$, i.e., 97.65792%
- This is scary but **polynomial time**

Research goal: Understanding the complexity of PQE

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

Research goal: Understanding the complexity of PQE

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

$\rightarrow$ Note that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

Research goal: Understanding the complexity of PQE

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

$\rightarrow$ Note that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

My work: several dichotomies on the PQE problem:

Research goal: Understanding the complexity of PQE

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

$\rightarrow$ Note that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

My work: several dichotomies on the PQE problem:

- **Existing results** on UCQ:
  - $\mathrm{PQE}(Q)$ is in #P for any UCQ Q and is **#P-hard** for some CQs
  - **Dichotomy** by Dalvi and Suciu: $\mathrm{PQE}(Q)$ for a UCQ Q is either **#P-hard** or **PTIME**

Research goal: Understanding the complexity of PQE

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

→ Note that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

My work: several dichotomies on the PQE problem:

- **Existing results** on UCQ:
  - $\mathrm{PQE}(Q)$ is in #P for any UCQ Q and is **#P-hard** for some CQs
  - **Dichotomy** by Dalvi and Suciu: $\mathrm{PQE}(Q)$ for a UCQ Q is either **#P-hard** or **PTIME**
- **This talk:** dichotomy on **homomorphism-closed queries**
  - $\mathrm{PQE}(Q)$ is **#P-hard** for all homomorphism-closed queries not equivalent to a safe UCQ

Research goal: Understanding the complexity of PQE

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

$\rightarrow$ Note that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

My work: several dichotomies on the PQE problem:

- **Existing results** on UCQ:
  - $\mathrm{PQE}(Q)$ is in #P for any UCQ Q and is **#P-hard** for some CQs
  - **Dichotomy** by Dalvi and Suciu: $\mathrm{PQE}(Q)$ for a UCQ Q is either **#P-hard** or **PTIME**
- **This talk:** dichotomy on **homomorphism-closed queries**
  - $\mathrm{PQE}(Q)$ is **#P-hard** for all homomorphism-closed queries not equivalent to a safe UCQ
- I'll also mention some of my work on **restricted graph classes**

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Basic complexity results

- Whenever we can **evaluate Q in PTIME**, then $\mathrm{PQE}(Q)$ is in **#P**

Basic complexity results

- Whenever we can **evaluate Q in PTIME**, then $\mathrm{PQE}(Q)$ is in **#P**
  - **#P:** counting class of problems expressible as the **number of accepting paths** of a nondeterministic polynomial-time Turing Machine

$\rightarrow$ Nondeterministically guess a possible world, then test the query

$\rightarrow$ In particular, $\mathrm{PQE}(Q)$ is in **#P** for any **UCQ** Q

Basic complexity results

- Whenever we can **evaluate Q in PTIME**, then $\mathrm{PQE}(Q)$ is in **#P**
  - **#P:** counting class of problems expressible as the **number of accepting paths** of a nondeterministic polynomial-time Turing Machine
  - $\rightarrow$ Nondeterministically guess a possible world, then test the query
  - $\rightarrow$ In particular, $\mathrm{PQE}(Q)$ is in **#P** for any **UCQ** Q
- For some queries Q, the task $\mathrm{PQE}(Q)$ is in **PTIME**

Basic complexity results

- Whenever we can **evaluate Q in PTIME**, then $\mathrm{PQE}(Q)$ is in **#P**
  - **#P:** counting class of problems expressible as the **number of accepting paths** of a nondeterministic polynomial-time Turing Machine

  $\rightarrow$ Nondeterministically guess a possible world, then test the query

  $\rightarrow$ In particular, $\mathrm{PQE}(Q)$ is in **#P** for any **UCQ** Q

- For some queries Q, the task $\mathrm{PQE}(Q)$ is in **PTIME**

  $\rightarrow$ e.g., $\exists x\, y \quad x \longrightarrow y$ or $\exists x\, y\, z \quad x \longrightarrow y \longrightarrow z$

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is #P-hard for the CQ Q :

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

PQE is sometimes #P-hard

Let us show that $\text{PQE}(Q)$ is #P-hard for the CQ $Q: x \longrightarrow y \longrightarrow z \longrightarrow w$

- 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

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is #P-hard for the CQ 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:

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ 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 $Y_1, \ldots, Y_m$

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is #P-hard for the CQ 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 $X_i \land Y_j$

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ 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: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ 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: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

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

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

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

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

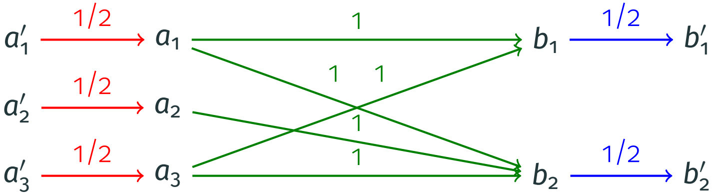
- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already **#P-hard** for so-called **PP2DNF formulas**:
  - **Positive** (no negation) and **Partitioned variables**: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

PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ 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: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

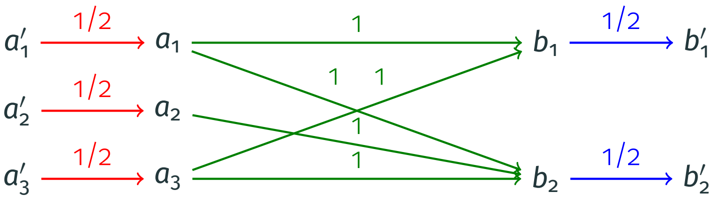


PQE is sometimes #P-hard

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ 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: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$



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

The “small” Dalvi and Suciu dichotomy

- Self-join-free CQ: only one edge of each color (no repeated color)

The “small” Dalvi and Suciu dichotomy

- Self-join-free CQ: only one edge of each color (no repeated color)

Theorem (Dalvi and Suciu, see Dalvi and Suciu 2007)

Let Q be a self-join-free CQ:

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

The "small" Dalvi and Suciu dichotomy

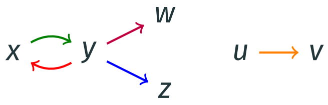
- **Self-join-free CQ**: only one edge of each color (no repeated color)

Theorem (Dalvi and Suciu, see Dalvi and Suciu 2007)

Let Q be a self-join-free CQ:

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

- A **star** is a CQ where each connected component has a **separator variable** that occurs in every edge of the component



The "small" Dalvi and Suciu dichotomy

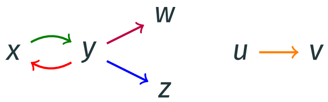
- Self-join-free CQ: only one edge of each color (no repeated color)

Theorem (Dalvi and Suciu, see Dalvi and Suciu 2007)

Let Q be a self-join-free CQ:

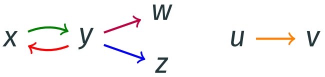
- *If Q is a star, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

- A star is a CQ where each connected component has a separator variable that occurs in every edge of the component



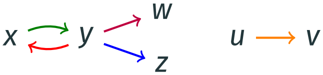
- The dichotomy generalizes to higher-arity data (hierarchical queries)

Proving the small dichotomy (upper bound)

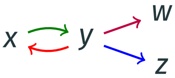


How to solve $\mathrm{PQE}(Q)$ for Q a self-join-free star?

Proving the small dichotomy (upper bound)



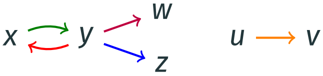
How to solve $\mathrm{PQE}(Q)$ for Q a self-join-free star?



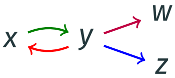
- We consider each connected component separately

$\rightarrow$ **Independent conjunction** over the connected components

Proving the small dichotomy (upper bound)

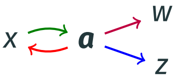


How to solve $\mathrm{PQE}(Q)$ for Q a self-join-free star?



- We consider each connected component separately

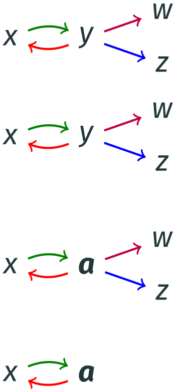
$\rightarrow$ **Independent conjunction** over the connected components



- We can test all possible values of the **separator variable**

$\rightarrow$ **Independent disjunction** over the values of the separator

Proving the small dichotomy (upper bound)



How to solve $\mathrm{PQE}(Q)$ for Q a self-join-free star?

- We consider each connected component separately

$\rightarrow$ **Independent conjunction** over the connected components

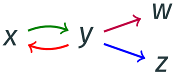
- We can test all possible values of the **separator variable**

$\rightarrow$ **Independent disjunction** over the values of the separator

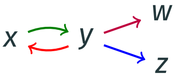
- For every match, we consider every **other variable** separately

$\rightarrow$ **Independent conjunction** over the variables

Proving the small dichotomy (upper bound)

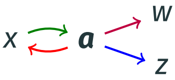


How to solve $\mathrm{PQE}(Q)$ for Q a self-join-free star?



- We consider each connected component separately

$\rightarrow$ **Independent conjunction** over the connected components



- We can test all possible values of the **separator variable**

$\rightarrow$ **Independent disjunction** over the values of the separator

- For every match, we consider every **other variable** separately

$\rightarrow$ **Independent conjunction** over the variables

- We consider every value for the **other variable**

$\rightarrow$ **Independent disjunction** over the possible assignments

$\rightarrow$ **Independent conjunction** over the facts

Proving the small dichotomy (lower bound)

Every non-star self-join-free CQ contains a pattern essentially like:

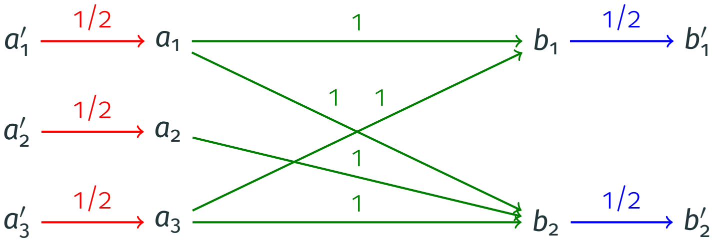
$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

Proving the small dichotomy (lower bound)

Every **non-star** self-join-free CQ contains a pattern essentially like:

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

We can use this to reduce from #SAT like before:



The "big" Dalvi and Suciu dichotomy

Full dichotomy on the **unions of conjunctive queries** (UCQs):

Theorem (Dalvi and Suciu 2012)

Let Q be a UCQ:

- *If Q is handled by a complicated algorithm $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

The “big” Dalvi and Suciu dichotomy

Full dichotomy on the **unions of conjunctive queries** (UCQs):

Theorem (Dalvi and Suciu 2012)

Let Q be a UCQ:

- *If Q is handled by a complicated algorithm $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

This result is **far more complicated** (but still generalizes to higher arity)

- **Upper bound:**
  - an algorithm generalizing the previous case with **inclusion-exclusion**
  - many **unpleasant details** (e.g., a ranking transformation)
- **Lower bound:** hardness proof on minimal cases where the algorithm does not work

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Going to more general queries

The case of ***UCQs*** *is settled! but what about* ***more expressive queries****?*

Going to more general queries

The case of ***UCQs*** *is settled! but what about* ***more expressive queries****?*

- Work by Fink and Olteanu 2016 about **negation**
- Some work on **ontology-mediated query answering** (Jung and Lutz 2012)

Going to more general queries

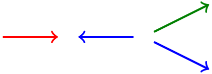
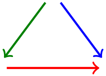
*The case of **UCQs** is settled! but what about **more expressive queries**?*

- Work by Fink and Olteanu 2016 about **negation**
- Some work on **ontology-mediated query answering** (Jung and Lutz 2012)

We study the case of **queries closed under homomorphisms**

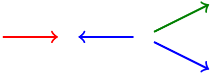
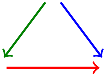
Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

Homomorphism-closed queries

- A homomorphism from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

- Homomorphism-closed query Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q

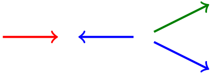
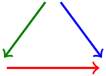
Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.

Homomorphism-closed queries

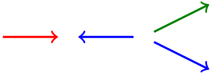
- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

 has a homomorphism to 

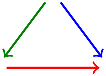
- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges



has a homomorphism to



- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed
- Homomorphism-closed queries can equivalently be seen as **infinite unions of CQs** (corresponding to their models)

Our result

We show:

Theorem (Amarilli and Ceylan 2020)

For any query Q closed under homomorphisms:

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

Our result

We show:

Theorem (Amarilli and Ceylan 2020)

For any query Q closed under homomorphisms:

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

- The same holds for RPQs, Datalog queries, etc.

Our result

We show:

Theorem (Amarilli and Ceylan 2020)

For any query Q closed under homomorphisms:

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

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\longrightarrow (\longrightarrow)^* \longrightarrow$

Our result

We show:

Theorem (Amarilli and Ceylan 2020)

For any query Q closed under homomorphisms:

- *Either* $\boldsymbol{Q}$ *is equivalent to a tractable UCQ and* $\mathrm{PQE}(\boldsymbol{Q})$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(\boldsymbol{Q})$ *is #P-hard*

- The same holds for RPQs, Datalog queries, 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 show:

Theorem (Amarilli and Ceylan 2020)

For any query Q closed under homomorphisms:

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

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\color{red}{\longrightarrow}\,(\color{green}{\longrightarrow})^*\color{blue}{\longrightarrow}$
  - It is not equivalent to a UCQ: infinite disjunction $\color{red}{\longrightarrow}\,(\color{green}{\longrightarrow})^i\color{blue}{\longrightarrow}$ for all $i \in \mathbb{N}$
  - Hence, $\mathrm{PQE}(Q)$ is #P-hard

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Going back to more restricted instances

OK, PQE *is* ***intractable*** *for essentially all queries. What now?*

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

$\rightarrow$ Could the same be true in the **probabilistic case**?

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

→ Could the same be true in the **probabilistic case**?

Theorem (Amarilli, Bourhis, and Senellart 2015; Amarilli, Bourhis, and Senellart 2016)

*Let $k \in \mathbb{N}$ be a constant bound, and let Q be a Boolean **monadic second-order** query. Then $\mathrm{PQE}(Q)$ is in **PTIME** on input TID instances with **treewidth** $\leq k$*

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

$\rightarrow$ Could the same be true in the **probabilistic case**?

Theorem (Amarilli, Bourhis, and Senellart 2015; Amarilli, Bourhis, and Senellart 2016)

*Let $k \in \mathbb{N}$ be a constant bound, and let Q be a Boolean **monadic second-order** query. Then $\mathrm{PQE}(Q)$ is in **PTIME** on input TID instances with **treewidth** $\leq k$*

*Conversely, there is a query Q for which $\mathrm{PQE}(Q)$ is intractable on **any** input instance family of unbounded treewidth (under some technical assumptions)*

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the subgraph counting (SC) problem:
  - $\rightarrow$ $\mathrm{SC}(Q)$: given a graph, how many of its subgraphs satisfy Q

Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the **subgraph counting** (SC) problem:
  - $\rightarrow$ $\mathrm{SC}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The SC problem **reduces** to PQE, but no obvious reduction in the other direction

Problem statement

What if we restricted probabilities on input instances to always be $1/2$?

- The PQE problem becomes the **subgraph counting** (SC) problem:
  → $\mathrm{SC}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The SC problem **reduces** to PQE, but no obvious reduction in the other direction

We study to **self-join-free CQs** and extend the "small" Dalvi and Suciu dichotomy to SC:

Theorem (Amarilli and Kimelfeld 2020)

Let Q be a self-join-free CQ:

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

→ This also extends **beyond arity two** (hierarchical queries)

Table of contents

Introduction and problem statement

Existing results

Main result: Dichotomy on homomorphism-closed queries

More restricted instances: Words, trees and bounded treewidth (1 slide)

More restricted instances: Unweighted instances (1 slide)

Conclusion and open problems

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Conclusion and open problems

We have seen:

- PQE is #P-hard for all homomorphism-closed queries except safe UCQs
- PQE is in PTIME for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like unweighted subgraph counting for self-join-free CQs

Future directions:

- Understanding tractable UCQs better, especially the connection to circuits

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Future directions:

- Understanding **tractable UCQs** better, especially the connection to **circuits**
- Tractable **approximation algorithms**, especially for recursive queries

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Future directions:

- Understanding **tractable UCQs** better, especially the connection to **circuits**
- Tractable **approximation algorithms**, especially for recursive queries
- Understand **unweighted subgraph counting** for more general classes

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Future directions:

- Understanding **tractable UCQs** better, especially the connection to **circuits**
- Tractable **approximation algorithms**, especially for recursive queries
- Understand **unweighted subgraph counting** for more general classes
- Extending to **arbitrary-arity data**

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Future directions:

- Understanding **tractable UCQs** better, especially the connection to **circuits**
- Tractable **approximation algorithms**, especially for recursive queries
- Understand **unweighted subgraph counting** for more general classes
- Extending to **arbitrary-arity data**
- Other query features: negation, inequalities, etc.

Conclusion and open problems

We have seen:

- PQE is **#P-hard** for all homomorphism-closed queries except safe UCQs
- PQE is **in PTIME** for MSO on bounded-treewidth graphs and intractable otherwise
- PQE behaves like **unweighted subgraph counting** for self-join-free CQs

Future directions:

- Understanding **tractable UCQs** better, especially the connection to **circuits**
- Tractable **approximation algorithms**, especially for recursive queries
- Understand **unweighted subgraph counting** for more general classes
- Extending to **arbitrary-arity data**
- Other query features: negation, inequalities, etc.
- Connections to other problems, especially **enumeration** of query results and **maintenance under updates**

Advertisement: TCS4F and “No free view? No review!”

Are you concerned about how academic research in theoretical computer science is contributing to the climate crisis?

If so, **sign the TCS4F pledge**! (Theoretical Computer Scientists 4 Future)

`www.tcs4f.org`

(with Thomas Schwentick, Thomas Colcombet, Hugo Férée)

Advertisement: TCS4F and “No free view? No review!”

Are you concerned about how academic research in theoretical computer science is contributing to the climate crisis?

If so, **sign the TCS4F pledge**! (Theoretical Computer Scientists 4 Future)

`www.tcs4f.org`

(with Thomas Schwentick, Thomas Colcombet, Hugo Férée)

TCS4F
THEORETICAL COMPUTER SCIENTISTS FOR FUTURE

NO FREE VIEW?
NO REVIEW!

Are you tired of doing reviewing work for conferences and journals that do not publish their research online?

If so, **sign the pledge “No free view? No review!”**

`www.nofreeviewnoreview.org`

(with Antonin Delpeuch)

Advertisement: TCS4F and “No free view? No review!”

Are you concerned about how academic research in theoretical computer science is contributing to the climate crisis?

If so, **sign the TCS4F pledge**! (Theoretical Computer Scientists 4 Future)

`www.tcs4f.org`

(with Thomas Schwentick, Thomas Colcombet, Hugo Férée)

TCS4F
THEORETICAL
COMPUTER
SCIENTISTS
FOR FUTURE

NO FREE VIEW?
NO REVIEW!

Are you tired of doing reviewing work for conferences and journals that do not publish their research online?

If so, **sign the pledge “No free view? No review!”**

`www.nofreeviewnoreview.org`

(with Antonin Delpeuch)

Thanks for your attention!

Amarilli, Antoine, Pierre Bourhis, and Pierre Senellart (2015). "Provenance Circuits for Trees and Treelike Instances". In: *ICALP*.

— (2016). "Tractable Lineages on Treelike Instances: Limits and Extensions". In: *PODS*.

Amarilli, Antoine and Ismail Ilkan Ceylan (2020). "A Dichotomy for Homomorphism-Closed Queries on Probabilistic Graphs". In: *ICDT*.

Amarilli, Antoine and Benny Kimelfeld (2020). "Uniform Reliability of Self-Join-Free Conjunctive Queries". Preprint: `https://arxiv.org/abs/1908.07093`.

Dalvi, Nilesh and Dan Suciu (2007). "The dichotomy of conjunctive queries on probabilistic structures". In: *Proc. PODS*.

— (2012). "The dichotomy of probabilistic inference for unions of conjunctive queries". In: *J. ACM* 59.6.

Bibliography ii

Fink, Robert and Dan Olteanu (2016). "Dichotomies for queries with negation in probabilistic databases". In: 41.1, 4:1–4:47.

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