Uniform Reliability of Self-Join-Free Conjunctive Queries

Antoine Amarilli[1], Benny Kimelfeld[2]

March 23, 2021

[1]Télécom Paris

[2]Technion

Uniform reliability

We study the uniform reliability (UR) problem for relational databases:

Uniform reliability

We study the uniform reliability (UR) problem for relational databases:

- Fix a Boolean Conjunctive Query (CQ) Q
- Define the counting problem $\mathsf{UR}(Q)$:

Uniform reliability

We study the uniform reliability (UR) problem for relational databases:

- Fix a Boolean Conjunctive Query (CQ) Q
- Define the counting problem $\mathrm{UR}(Q)$:
  - Input: a database instance I
  - Output: how many subinstances of I (subset of facts) satisfy Q

Uniform reliability

We study the **uniform reliability** (UR) problem for relational databases:

- Fix a **Boolean Conjunctive Query** (CQ) Q
- Define the **counting problem** UR(Q):
  - **Input:** a database instance I
  - **Output:** how many **subinstances** of I (subset of facts) satisfy Q

Class

Class	Room	Date
CS 1	101	2021-03-26
CS 2	101	2021-04-02

Lockdown

Date
2021-03-26
2021-04-02

Consider the query:

$Q : \exists c\, r\, d\ \text{Class}(c, r, d) \wedge \text{Lockdown}(d)$

Uniform reliability

We study the uniform reliability (UR) problem for relational databases:

- Fix a Boolean Conjunctive Query (CQ) Q
- Define the counting problem $\text{UR}(Q)$:
  - Input: a database instance I
  - Output: how many subinstances of I (subset of facts) satisfy Q

Class

Class	Room	Date
CS 1	101	2021-03-26
CS 2	101	2021-04-02

Lockdown

Date
2021-03-26
2021-04-02

Consider the query:

$Q : \exists c\, r\, d\ \text{Class}(c, r, d) \wedge \text{Lockdown}(d)$

The number of subinstances satisfying Q is:

Uniform reliability

We study the **uniform reliability** (UR) problem for relational databases:

- Fix a **Boolean Conjunctive Query** (CQ) Q
- Define the **counting problem** UR(Q):
  - **Input:** a database instance I
  - **Output:** how many **subinstances** of I (subset of facts) satisfy Q

Class

Class	Room	Date
CS 1	101	2021-03-26
CS 2	101	2021-04-02

Lockdown

Date
2021-03-26
2021-04-02

Consider the query:

$Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \wedge \mathrm{Lockdown}(d)$

The number of subinstances satisfying Q is: $0 + 2 + 2 + 3 = 7$

Uniform reliability

We study the **uniform reliability** (UR) problem for relational databases:

- Fix a **Boolean Conjunctive Query** (CQ) Q
- Define the **counting problem** $\mathsf{UR}(Q)$:
  - **Input:** a database instance I
  - **Output:** how many **subinstances** of I (subset of facts) satisfy Q

Class

Class	Room	Date
CS 1	101	2021-03-26
CS 2	101	2021-04-02

Lockdown

Date
2021-03-26
2021-04-02

Consider the query:

$Q : \exists c\, r\, d\ \mathsf{Class}(c, r, d) \wedge \mathsf{Lockdown}(d)$

The number of subinstances satisfying Q is: $0 + 2 + 2 + 3 = 7$

- We can always solve $\mathsf{UR}(Q)$ by looking at **all subinstances** (exponential)

$\rightarrow$ **When can we achieve a better complexity?**

Motivation and connections

- We think that uniform reliability is a natural question

Motivation and connections

- We think that uniform reliability is a natural question
- It also connects to other works:

Motivation and connections

- We think that uniform reliability is a natural question
- It also connects to other works:
  - Query explanations using measures from computational social choice:
    - the Shapley value [Livshits et al., 2020]
    - the causal effect [Salimi, 2016]

Motivation and connections

- We think that uniform reliability is a natural question
- It also connects to other works:
  - Query explanations using measures from computational social choice:
    - the Shapley value [Livshits et al., 2020]
    - the causal effect [Salimi, 2016]
  - Probabilistic query evaluation on tuple-independent databases [Suciu et al., 2011]

Motivation and connections

- We think that uniform reliability is a natural question
- It also connects to other works:
  - Query explanations using measures from computational social choice:
    - the Shapley value [Livshits et al., 2020]
    - the causal effect [Salimi, 2016]
  - Probabilistic query evaluation on tuple-independent databases [Suciu et al., 2011]

→ Let's review the line of work on probabilistic query evaluation

Uncertain data and tuple-independent databases (TID)

- We consider data in the **relational model** on which we have **uncertainty**
- **Tuple-independent databases** (TID): the simplest probabilistic model

Class

Class	Room	Date	
CS 1	101	2021-03-26	80%
CS 2	101	2021-04-02	70%

Lockdown

Date	
2021-03-26	20%
2021-04-02	40%

Uncertain data and tuple-independent databases (TID)

- We consider data in the **relational model** on which we have **uncertainty**
- **Tuple-independent databases** (TID): the simplest probabilistic model

Class

Class	Room	Date	
CS 1	101	2021-03-26	80%
CS 2	101	2021-04-02	70%

Lockdown

Date	
2021-03-26	20%
2021-04-02	40%

- Semantics:
  - Every tuple is annotated with a **probability**
  - We assume that all tuples are **independent**
  - A TID concisely represents a **probability distribution** over the **subinstances**

Uncertain data and tuple-independent databases (TID)

- We consider data in the **relational model** on which we have **uncertainty**
- **Tuple-independent databases** (TID): the simplest probabilistic model

Class

Class	Room	Date	
CS 1	101	2021-03-26	80%
CS 2	101	2021-04-02	70%

Lockdown

Date	
2021-03-26	20%
2021-04-02	40%

- Semantics:
  - Every tuple is annotated with a **probability**
  - We assume that all tuples are **independent**
  - A TID concisely represents a **probability distribution** over the **subinstances**

→ **Uniform reliability** amounts to a TID where **all facts have probability 1/2**

Probabilistic query evaluation (PQE)

- We consider again Boolean Conjunctive Queries (CQs)
  - e.g., $Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \wedge \mathrm{Lockdown}(d)$

Probabilistic query evaluation (PQE)

- We consider again **Boolean Conjunctive Queries** (CQs)
  - e.g., $Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \land \mathrm{Lockdown}(d)$
- **Semantics** of Q on a TID: compute the **probability** that Q is true

Probabilistic query evaluation (PQE)

- We consider again Boolean Conjunctive Queries (CQs)
  - e.g., $Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \land \mathrm{Lockdown}(d)$
- Semantics of Q on a TID: compute the probability that Q is true
- Formally, problem PQE(Q) for a fixed CQ Q:
  - Input: a TID I
  - Output: the total probability of the subinstances of I where Q is true

Probabilistic query evaluation (PQE)

- We consider again **Boolean Conjunctive Queries** (CQs)
  - e.g., $Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \wedge \mathrm{Lockdown}(d)$
- **Semantics** of Q on a TID: compute the **probability** that Q is true
- Formally, problem **PQE(Q)** for a fixed CQ Q:
  - **Input:** a TID I
  - **Output:** the total probability of the subinstances of I where Q is true
- Again we can solve **PQE(Q)** by looking at all subinstances (exponential)

Probabilistic query evaluation (PQE)

- We consider again **Boolean Conjunctive Queries** (CQs)
  - e.g., $Q : \exists c\, r\, d\ \mathrm{Class}(c, r, d) \wedge \mathrm{Lockdown}(d)$
- **Semantics** of Q on a TID: compute the **probability** that Q is true
- Formally, problem **PQE(Q)** for a fixed CQ Q:
  - **Input:** a TID I
  - **Output:** the total probability of the subinstances of I where Q is true
- Again we can solve **PQE(Q)** by looking at all subinstances (exponential)

→ **When can we achieve a better complexity?**

Existing results

- Complexity of PQE shown in [Dalvi and Suciu, 2007] for **self-join-free CQs** (SJFCQs)
  - A CQ is **self-join-free** if no relation symbol is repeated
- Later extended to **unions of conjunctive queries** [Dalvi and Suciu, 2012]

Existing results

- Complexity of PQE shown in [Dalvi and Suciu, 2007] for **self-join-free CQs** (SJFCQs)
    - A CQ is **self-join-free** if no relation symbol is repeated
- Later extended to **unions of conjunctive queries** [Dalvi and Suciu, 2012]

In this work we stick to the result on SJFCQs:

Theorem [Dalvi and Suciu, 2007]

Let **Q** *be a SJFCQ. Then:*

- *Either* **Q** *is* ***hierarchical*** *and* PQE(**Q**) *is in PTIME*
- *Or* **Q** *is* ***not hierarchical*** *and* PQE(**Q**) *is #P-hard*

Existing results

- Complexity of PQE shown in [Dalvi and Suciu, 2007] for **self-join-free CQs** (SJFCQs)
  - A CQ is **self-join-free** if no relation symbol is repeated
- Later extended to **unions of conjunctive queries** [Dalvi and Suciu, 2012]

In this work we stick to the result on SJFCQs:

Theorem [Dalvi and Suciu, 2007]

Let **Q** *be a SJFCQ. Then:*

- *Either* **Q** *is* ***hierarchical*** *and* PQE(**Q**) *is in* *PTIME*
- *Or* **Q** *is* ***not hierarchical*** *and* PQE(**Q**) *is* *#**P**-hard*

What is this class of **hierarchical CQs**?

Hierarchical CQs

For a CQ Q, write **atoms**(x) for the set of atoms where x appears

- A CQ is hierarchical if for every variables x and y
  - Either **atoms**(x) and **atoms**(y) are disjoint
  - Or one is included in the other

Hierarchical CQs

For a CQ Q, write **atoms**(x) for the set of atoms where x appears

- A CQ is **hierarchical** if for every variables x and y
  - Either **atoms**(x) and **atoms**(y) are **disjoint**
  - Or one is **included** in the other

- A CQ is **non-hierarchical** if there are two variables x and y such that
  - Some atom contains **both x and y**
  - Some atom contains **x but not y**
  - Some atom contains **y but not x**

  → Simplest example: the **R-S-T query**: $Q_1 : \exists x\, y\; R(x), S(x, y), T(y)$

Hierarchical CQs

For a CQ Q, write **atoms**(x) for the set of atoms where x appears

- A CQ is **hierarchical** if for every variables x and y
  - Either **atoms**(x) and **atoms**(y) are **disjoint**
  - Or one is **included** in the other

- A CQ is **non-hierarchical** if there are two variables x and y such that
  - Some atom contains **both x and y**
  - Some atom contains **x but not y**
  - Some atom contains **y but not x**

  → Simplest example: the ***R-S-T* query**: $Q_1 : \exists x\, y\ R(x), S(x,y), T(y)$

- **Intuition** for arity-2 queries: the hierarchical CQs are **unions of star-shaped patterns**

Hierarchical CQs

For a CQ Q, write **atoms**(x) for the set of atoms where x appears

- A CQ is **hierarchical** if for every variables x and y
  - Either **atoms**(x) and **atoms**(y) are **disjoint**
  - Or one is **included** in the other

- A CQ is **non-hierarchical** if there are two variables x and y such that
  - Some atom contains **both x and y**
  - Some atom contains **x but not y**
  - Some atom contains **y but not x**

  → Simplest example: the ***R*-*S*-*T* query**: $Q_1 : \exists x\, y\; R(x), S(x,y), T(y)$

- **Intuition** for arity-2 queries: the hierarchical CQs are **unions of star-shaped patterns**

Exercise: Is our example CQ hierarchical? $\exists c\, r\, d\; \mathrm{Class}(c,r,d) \wedge \mathrm{Lockdown}(d)$

Hierarchical CQs

For a CQ Q, write **atoms**(x) for the set of atoms where x appears

- A CQ is **hierarchical** if for every variables x and y
  - Either **atoms**(x) and **atoms**(y) are **disjoint**
  - Or one is **included** in the other
- A CQ is **non-hierarchical** if there are two variables x and y such that
  - Some atom contains **both x and y**
  - Some atom contains **x but not y**
  - Some atom contains **y but not x**

  → Simplest example: the ***R*-*S*-*T* query**: $Q_1 : \exists x\, y\; R(x), S(x, y), T(y)$
- **Intuition** for arity-2 queries: the hierarchical CQs are **unions of star-shaped patterns**

Exercise: Is our example CQ hierarchical? $\exists c\, r\, d\; \mathrm{Class}(c, r, d) \wedge \mathrm{Lockdown}(d)$ ... **Yes!**

Our results on uniform reliability

Let us return to our problem of uniform reliability (UR):

Our results on uniform reliability

Let us return to our problem of uniform reliability (UR):

- We only study the problem on SJFCQs (see future work)

Our results on uniform reliability

Let us return to our problem of uniform reliability (UR):

- We only study the problem on SJFCQs (see future work)
- For hierarchical SJFCQs, UR is tractable because PQE is tractable

Our results on uniform reliability

Let us return to our problem of **uniform reliability** (UR):

- We only study the problem on **SJFCQs** (see future work)
- For **hierarchical SJFCQs**, UR is tractable because PQE is tractable
- For **non-hierarchical SJFCQs**, the complexity of UR is **unknown**
  - The **hardness proof** of PQE (see later) crucially uses probabilities **1/2** and **1**

Our results on uniform reliability

Let us return to our problem of **uniform reliability** (UR):

- We only study the problem on **SJFCQs** (see future work)
- For **hierarchical SJFCQs**, UR is tractable because PQE is tractable
- For **non-hierarchical SJFCQs**, the complexity of UR is **unknown**
  - The **hardness proof** of PQE (see later) crucially uses probabilities $1/2$ and 1

We settle the complexity of UR for SJFCQs by showing:

Theorem

Let Q be a non-hierarchical SJFCQ. Then $\mathsf{UR}(Q)$ is #P-hard.

Our results on uniform reliability

Let us return to our problem of **uniform reliability** (UR):

- We only study the problem on **SJFCQs** (see future work)
- For **hierarchical SJFCQs**, UR is tractable because PQE is tractable
- For **non-hierarchical SJFCQs**, the complexity of UR is **unknown**
  - The **hardness proof** of PQE (see later) crucially uses probabilities $1/2$ and 1

We settle the complexity of UR for SJFCQs by showing:

Theorem

Let Q be a non-hierarchical SJFCQ. Then $\mathsf{UR}(Q)$ is #P-hard.

Rest of the talk: proof sketch of this result

Reducing to R-S-T-type queries

- An ***R*-*S*-*T*-type query** is a non-hierarchical SJFCQ of the form:

$$R_1(x), \dots, R_r(x), S_1(x,y), \dots, S_s(x,y), T_1(y), \dots, T_t(y)$$

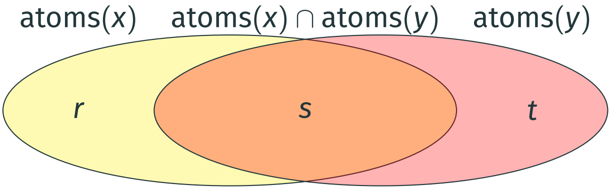
for some integers $r, s, t > 0$

Reducing to R-S-T-type queries

- An ***R*-*S*-*T*-type query** is a non-hierarchical SJFCQ of the form:

$$R_1(x), \ldots, R_r(x), S_1(x,y), \ldots, S_s(x,y), T_1(y), \ldots, T_t(y)$$

  for some integers $r, s, t > 0$
- **Lemma:** for any non-hierarchical SJFCQ Q, there is an R-S-T-type query Q' such that $\mathsf{UR}(Q')$ reduces to $\mathsf{UR}(Q)$



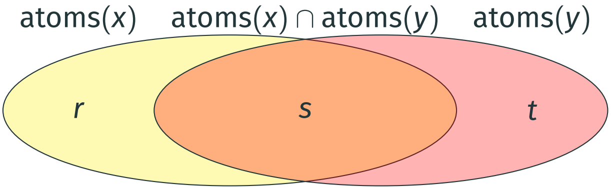
Reducing to R-S-T-type queries

- An ***R-S-T*-type query** is a non-hierarchical SJFCQ of the form:

$$R_1(x), \ldots, R_r(x), S_1(x,y), \ldots, S_s(x,y), T_1(y), \ldots, T_t(y)$$

  for some integers $r, s, t > 0$

- **Lemma:** for any non-hierarchical SJFCQ Q, there is an R-S-T-type query Q' such that $\mathrm{UR}(Q')$ reduces to $\mathrm{UR}(Q)$



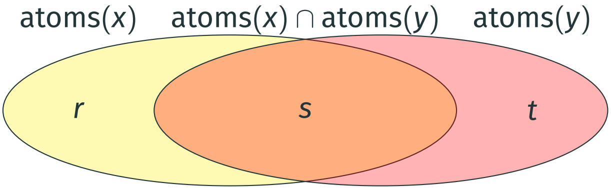
- So it suffices to show that $\mathrm{UR}(Q')$ is #P-hard for the R-S-T-type queries Q'

Reducing to R-S-T-type queries

- An ***R*-*S*-*T*-type query** is a non-hierarchical SJFCQ of the form:

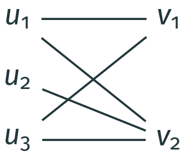
$$R_1(x), \ldots, R_r(x), S_1(x,y), \ldots, S_s(x,y), T_1(y), \ldots, T_t(y)$$

  for some integers $r, s, t > 0$
- **Lemma:** for any non-hierarchical SJFCQ Q, there is an R-S-T-type query Q' such that $\mathrm{UR}(Q')$ reduces to $\mathrm{UR}(Q)$



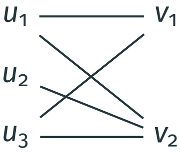
- So it suffices to show that $\mathrm{UR}(Q')$ is #P-hard for the R-S-T-type queries Q'
- **In this talk:** we focus for simplicity on $Q_1 : \exists x\, y\; R(x), S(x,y), T(y)$

Hard problem: counting independent sets of bipartite graphs



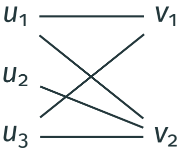
- **Independent set** of a bipartite graph: subset of its vertices that contains no edge
  - Example: $\{u_2, v_1\}$

Hard problem: counting independent sets of bipartite graphs



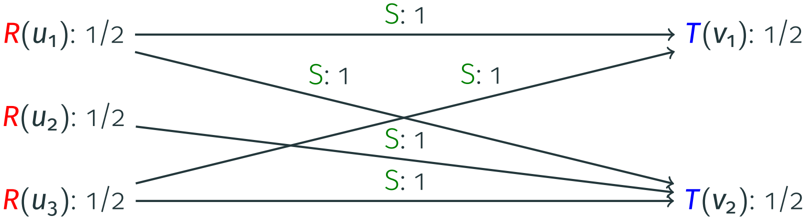
- Independent set of a bipartite graph: subset of its vertices that contains no edge
  - Example: $\{u_2, v_1\}$
- It is #P-hard, given a bipartite graph, to count its independent sets

Hard problem: counting independent sets of bipartite graphs

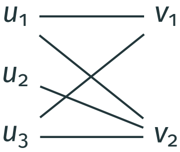


- Independent set of a bipartite graph: subset of its vertices that contains no edge
  - Example: $\{u_2, v_1\}$
- It is #P-hard, given a bipartite graph, to count its independent sets

This easily shows the #P-hardness of PQE (but not UR!) for $Q_1 : \exists x\, y\; R(x), S(x,y), T(y)$:

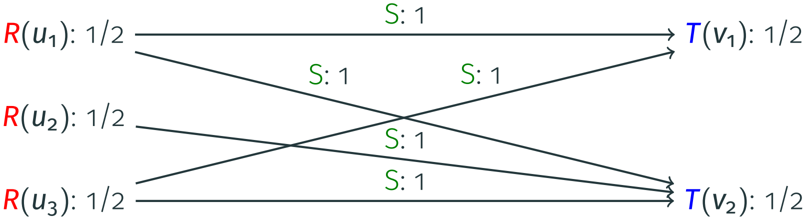


Hard problem: counting independent sets of bipartite graphs



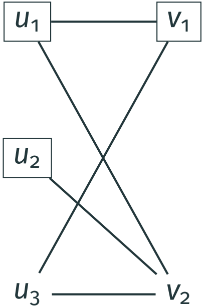
- **Independent set** of a bipartite graph: subset of its vertices that contains no edge
  - Example: $\{u_2, v_1\}$
- It is #P-hard, given a bipartite graph, to count its independent sets

This easily shows the #P-hardness of PQE (but not UR!) for $Q_1 : \exists x\, y\ R(x), S(x,y), T(y)$:



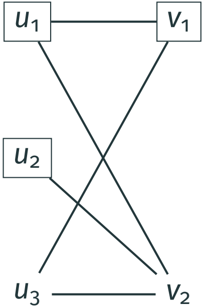
We will show how to reduce from counting independent sets to $\mathbf{UR}(Q_1)$

Idea: parameterizing the count



For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

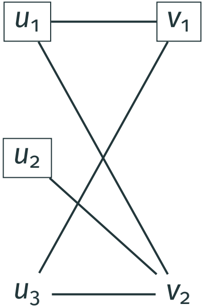
Idea: parameterizing the count



For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

- $\mathrm{c}(W)$ the number of edges **contained** in W
  - Here, $\mathrm{c}(W) = 1$

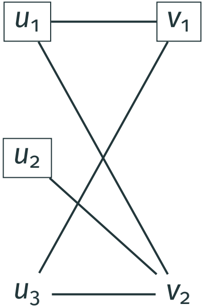
Idea: parameterizing the count



For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

- $\mathrm{c}(W)$ the number of edges **contained** in W
  - **Here**, $\mathrm{c}(W) = 1$
- $\mathrm{d}(W)$ (resp., $\mathrm{d}'(W)$) the number of edges having exactly their left (resp., right) endpoint in W
  - **Here**, $\mathrm{d}(W) = 2$ and $\mathrm{d}'(W) = 1$

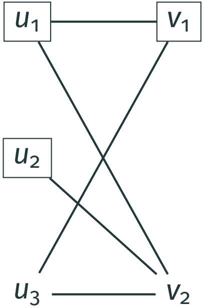
Idea: parameterizing the count



For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

- $\mathrm{c}(W)$ the number of edges **contained** in W
  - **Here**, $\mathrm{c}(W) = 1$
- $\mathrm{d}(W)$ (resp., $\mathrm{d}'(W)$) the number of edges having exactly their left (resp., right) endpoint in W
  - **Here**, $\mathrm{d}(W) = 2$ and $\mathrm{d}'(W) = 1$
- $\mathrm{e}(W)$ the number of edges **excluded from** W
  - **Here**, $\mathrm{e}(W) = 1$

Idea: parameterizing the count

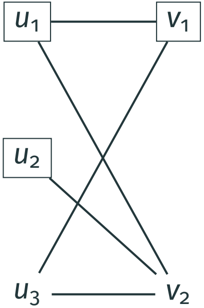


For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

- $\mathrm{c}(W)$ the number of edges **contained** in W
  - **Here**, $\mathrm{c}(W) = 1$
- $\mathrm{d}(W)$ (resp., $\mathrm{d}'(W)$) the number of edges having exactly their left (resp., right) endpoint in W
  - **Here**, $\mathrm{d}(W) = 2$ and $\mathrm{d}'(W) = 1$
- $\mathrm{e}(W)$ the number of edges **excluded from** W
  - **Here**, $\mathrm{e}(W) = 1$

- **Hard problem:** counting independent sets $X = |\{W \subseteq U \cup V \mid c(W) = 0\}|$

Idea: parameterizing the count



For a bipartite graph (U, V, E) and a subset $W \subseteq U \cup V$ of vertices, write:

- $\mathrm{c}(W)$ the number of edges **contained** in W
  - Here, $\mathrm{c}(W) = 1$
- $\mathrm{d}(W)$ (resp., $\mathrm{d}'(W)$) the number of edges having exactly their left (resp., right) endpoint in W
  - Here, $\mathrm{d}(W) = 2$ and $\mathrm{d}'(W) = 1$
- $\mathrm{e}(W)$ the number of edges **excluded from** W
  - Here, $\mathrm{e}(W) = 1$

- **Hard problem:** counting independent sets $X = |\{W \subseteq U \cup V \mid c(W) = 0\}|$
- **Harder problem:** computing all the values:

$$X_{c,d,d',e} = \left|\{W \subseteq U \cup V \mid \mathrm{c}(W) = c \text{ and } \mathrm{d}(W) = d \text{ and } \mathrm{d}'(W) = d' \text{ and } \mathrm{e}(W) = e\}\right|$$

Idea: coding to several copies

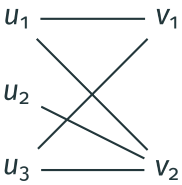
- We want to design a reduction:
  - We reduce from (we want): given a bipartite graph G, compute the $X_{c,d,d',e}$
  - We reduce to (we have): given a database instance D, compute $\mathrm{UR}(Q_1)$

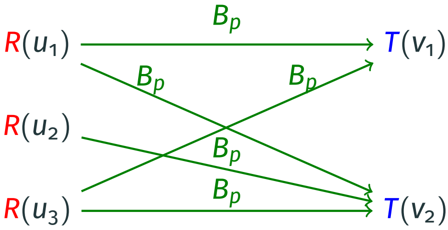
Idea: coding to several copies

- We want to design a **reduction**:
  - We reduce **from** (we want): given a bipartite graph G, compute the $X_{c,d,d',e}$
  - We reduce **to** (we have): given a database instance D, compute $\mathrm{UR}(Q_1)$
- **Idea:** code G to a **family** of instances D_p **indexed** by $p > 0$

Idea: coding to several copies

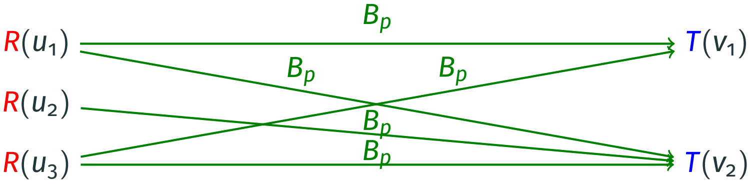
- We want to design a **reduction**:
  - We reduce **from** (we want): given a bipartite graph G, compute the $X_{c,d,d',e}$
  - We reduce **to** (we have): given a database instance D, compute $\mathrm{UR}(Q_1)$
- **Idea:** code G to a **family** of instances D_p **indexed** by $p > 0$
- Fix a **box** $B_p(a, b)$ for index $p > 0$: an instance with two distinguished elements (a, b)
- **Code** G for index $p > 0$ to an instance by:
  - putting an R-fact on each U-vertex and a T-fact on each V-vertex
  - coding every edge (u, v) by a **copy of the box** $B_p(u, v)$





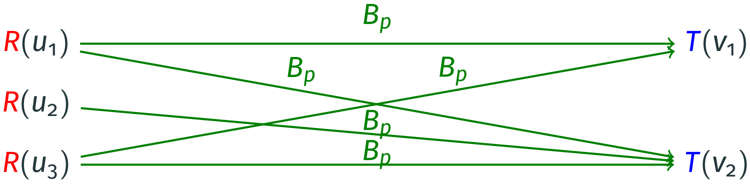
Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



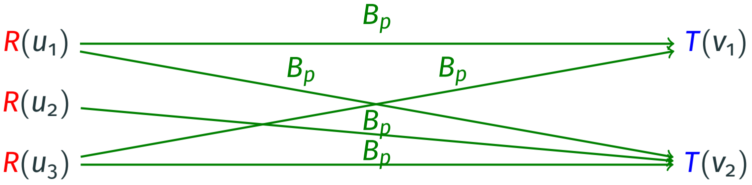
- We have:

$$N_p = \sum_{W \subseteq V} N_p^W$$

where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W

Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



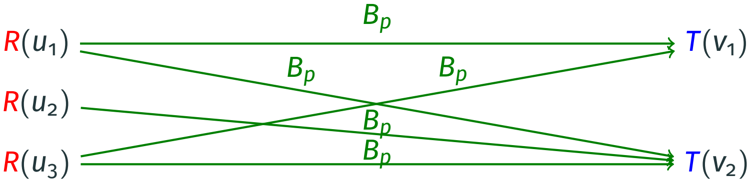
- We have:

$$N_p = \sum_{W \subseteq V} N_p^W$$

  where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W
- Now N_p^W only depends on:

Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



- We have:

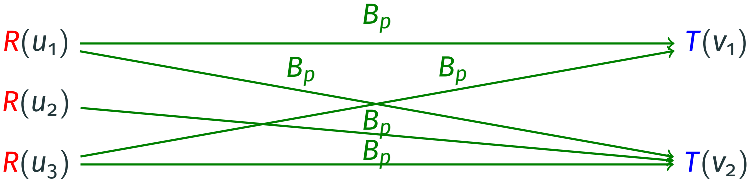
$$N_p = \sum_{W \subseteq V} N_p^W$$

where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W

- Now N_p^W only depends on:
  - The numbers $\mathrm{c}(W), \mathrm{d}(W), \mathrm{d}'(W), \mathrm{e}(W)$ of edges contained, dangling, or excluded from W

Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



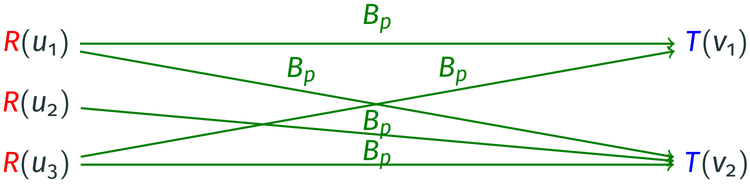
- We have:

$$N_p = \sum_{W \subseteq V} N_p^W$$

  where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W
- Now N_p^W only depends on:
  - The numbers $\mathrm{c}(W), \mathrm{d}(W), \mathrm{d}'(W), \mathrm{e}(W)$ of edges contained, dangling, or excluded from W
  - The numbers γ_p, δ_p, δ'_p, η_p of subinstances of the box B_p that violate Q_1 when fixing R-facts on a and/or T-facts on b

Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



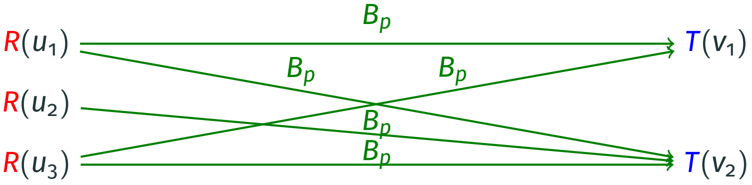
- We have:

$$N_p = \sum_{W \subseteq V} N_p^W = \sum_{W \subseteq V} \gamma_p^{\mathrm{c}(W)} \delta_p^{\mathrm{d}(W)} (\delta'_p)^{\mathrm{d}'(W)} \eta_p^{\mathrm{e}(W)}$$

  where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W
- Now N_p^W only depends on:
  - The numbers $\mathrm{c}(W), \mathrm{d}(W), \mathrm{d}'(W), \mathrm{e}(W)$ of edges contained, dangling, or excluded from W
  - The numbers γ_p, δ_p, δ'_p, η_p of subinstances of the box B_p that violate Q_1 when fixing R-facts on a and/or T-facts on b

Getting an equation system

Take the coding of G for index p, and compute the number N_p of subinstances violating Q_1



- We have:

$$N_p = \sum_{W \subseteq V} N_p^W = \sum_{W \subseteq V} \gamma_p^{\mathrm{c}(W)} \delta_p^{\mathrm{d}(W)} (\delta'_p)^{\mathrm{d}'(W)} \eta_p^{\mathrm{e}(W)} = \sum_{c,d,d',e} X_{c,d,d',e} \times \gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$$

  where N_p^W is the number of subinstances violating Q_1 when fixing the R-facts and T-facts to be precisely on W
- Now N_p^W only depends on:
  - The numbers $\mathrm{c}(W), \mathrm{d}(W), \mathrm{d}'(W), \mathrm{e}(W)$ of edges contained, dangling, or excluded from W
  - The numbers γ_p, δ_p, δ'_p, η_p of subinstances of the box B_p that violate Q_1 when fixing R-facts on a and/or T-facts on b

Equation system and conclusion

We have shown the equation:

$$N_p = \sum_{c,d,d',e} X_{c,d,d',e} \times \gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$$

where:

- The $X_{c,d,d',e}$ are what we **want** (to count independent sets)
- The N_p are what we **have** (by solving $\mathrm{UR}(Q_1)$)
- The $\gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$ are **coefficients** of a matrix M depending on the box family B_p

Equation system and conclusion

We have shown the equation:

$$N_p = \sum_{c,d,d',e} X_{c,d,d',e} \times \gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$$

where:

- The $X_{c,d,d',e}$ are what we **want** (to count independent sets)
- The N_p are what we **have** (by solving $\mathsf{UR}(Q_1)$)
- The $\gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$ are **coefficients** of a matrix M depending on the box family B_p

In other words we have:

$$\vec{N} = M\vec{X}$$

Equation system and conclusion

We have shown the equation:

$$N_p = \sum_{c,d,d',e} X_{c,d,d',e} \times \gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$$

where:

- The $X_{c,d,d',e}$ are what we want (to count independent sets)
- The N_p are what we have (by solving $\mathrm{UR}(Q_1)$)
- The $\gamma_p^c \delta_p^d (\delta'_p)^{d'} \eta_p^e$ are coefficients of a matrix M depending on the box family B_p

In other words we have:

$$\vec{N} = M\vec{X}$$

We find a box family where M is invertible, so we recover $\vec{X}$ from $\vec{N}$, showing hardness

Conclusion and future work

- We have shown that **uniform reliability (UR)** for non-hierarchical SJFCQs is #P-hard, so it is no easier than PQE
- We also have **preliminary results** for other PQE restrictions

Conclusion and future work

- We have shown that **uniform reliability (UR)** for non-hierarchical SJFCQs is #P-hard, so it is no easier than PQE
- We also have **preliminary results** for other PQE restrictions

Future work directions:

- Can we extend to the **UCQ dichotomy**, e.g., following [Kenig and Suciu, 2020]?
- What about the case of PQE with a **constant probability** $\neq 1/2$? or a different **constant probability per relation**?
- Which connection to **symmetric model counting** [Beame et al., 2015]?

Conclusion and future work

- We have shown that **uniform reliability (UR)** for non-hierarchical SJFCQs is #P-hard, so it is no easier than PQE
- We also have **preliminary results** for other PQE restrictions

Future work directions:

- Can we extend to the **UCQ dichotomy**, e.g., following [Kenig and Suciu, 2020]?
- What about the case of PQE with a **constant probability $\neq 1/2$**?
  or a different **constant probability per relation**?
- Which connection to **symmetric model counting** [Beame et al., 2015]?

Thanks for your attention!

References i

- Beame, P., Van den Broeck, G., Gribkoff, E., and Suciu, D. (2015).
  Symmetric weighted first-order model counting.
  In *PODS*.
- Dalvi, N. and Suciu, D. (2007).
  Efficient query evaluation on probabilistic databases.
  VLDB Journal, 16(4):523–544.
- Dalvi, N. and Suciu, D. (2012).
  The dichotomy of probabilistic inference for unions of conjunctive queries.
  J. ACM, 59(6).

Kenig, B. and Suciu, D. (2020).
A dichotomy for the generalized model counting problem for unions of conjunctive queries.
CoRR, abs/2008.00896.
To appear at PODS 2021.

Livshits, E., Bertossi, L., Kimelfeld, B., and Sebag, M. (2020).
The Shapley Value of Tuples in Query Answering.
In *ICDT*.

Salimi, B. (2016).
Quantifying Causal Effects on Query Answering in Databases.
In *TaPP*.

Suciu, D., Olteanu, D., Ré, C., and Koch, C. (2011).
Probabilistic Databases.
Synthesis Lectures on Data Management. Morgan & Claypool Publishers.