The Intensional-Extensional Problem in Probabilistic Databases

Mikaël Monet

October 16th, 2023

Probabilistic Circuits and Logic workshop

Inria

Simons Institute for the Theory of Computing

Outline

1. Recap from Dan Suciu's talk

    Tuple-independent probabilistic databases

    Provenance and knowledge compilation

    The Intensional-Extensional problem

2. Solving the problem for a specific class of UCQs

3. The non-cancelling intersections conjecture

Recap from Dan Suciu’s talk

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - $\rightarrow$ simplest formalism: tuple-independent database

$D =$

Likes		**p**
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - → simplest formalism: tuple-independent database

$D' =$

Likes		p
		0.5
Alice	John	1
		0.2
John	Bob	0.7

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty

  $\rightarrow$ simplest formalism: tuple-independent database

$D' =$

Likes		p
		0.5
Alice	John	1
		0.2
John	Bob	0.7

$\Pr(D') = (1 - 0.5) \times 1 \times (1 - 0.2) \times 0.7$

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - $\rightarrow$ simplest formalism: tuple-independent database

$D =$

Likes		**p**
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$q =$ "there are two people who like the same person"

$\exists x, y, z : L(x,z) \wedge L(y,z) \wedge x \neq y$

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - $\rightarrow$ simplest formalism: tuple-independent database

$D =$

Likes		p
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$q =$ "there are two people who like the same person"

$\exists x, y, z : L(x, z) \wedge L(y, z) \wedge x \neq y$

$\Pr(D \models q) = \sum_{\substack{D' \subseteq D \\ D' \models q}} \Pr(D')$

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - $\rightarrow$ simplest formalism: tuple-independent database

$D =$

Likes		**p**
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$q =$ "there are two people who like the same person"

$\exists x, y, z : L(x,z) \wedge L(y,z) \wedge x \neq y$

$\Pr(D \models q) = \sum_{\substack{D' \subseteq D \\ D' \models q}} \Pr(D')$ (**not efficient**)

Tuple-independent probabilistic databases

- **Probabilistic databases**: to represent data uncertainty
  - $\rightarrow$ simplest formalism: tuple-independent database

$D =$

Likes		**p**
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$q =$ "there are two people who like the same person"
$\exists x, y, z \; : L(x,z) \wedge L(y,z) \wedge x \neq y$

$$\Pr(D \models q) = 1 - \Big[(1-0.5)(1-0.2)(1-0.7) + 0.5(1-0.2)(1-0.7) + (1-0.5)0.2(1-0.7) + (1-0.5)(1-0.2)0.7\Big]$$

The probabilistic query evaluation problem ($\mathrm{PQE}(q)$)

Definition: problem $\mathrm{PQE}(q)$, for q a Boolean query

Input: a tuple-independent probabilistic database D

Output: $\Pr(D \models q)$

The probabilistic query evaluation problem ($\mathrm{PQE}(q)$)

Definition: problem $\mathrm{PQE}(q)$, for q a Boolean query

Input: a tuple-independent probabilistic database D

Output: $\Pr(D \models q)$

- Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the (data) complexity of $\mathrm{PQE}(q)$ for unions of conjunctive queries:
  - either $\mathrm{PQE}(q) \in \mathrm{PTIME}$, and q is called "safe"
  - or $\mathrm{PQE}(q)$ is $\mathrm{FP}^{\#\mathrm{P}}$**-hard**, and q is called "unsafe"

The probabilistic query evaluation problem ($\mathrm{PQE}(q)$)

Definition: problem $\mathrm{PQE}(q)$, for q a Boolean query

Input: a tuple-independent probabilistic database D

Output: $\Pr(D \models q)$

- Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the (data) complexity of $\mathrm{PQE}(q)$ for unions of conjunctive queries:
  - either $\mathrm{PQE}(q) \in \mathrm{PTIME}$, and q is called "safe"
  - or $\mathrm{PQE}(q)$ is $\mathrm{FP}^{\#\mathrm{P}}$**-hard**, and q is called "unsafe"
- Their algorithm for a safe query q essentially uses three rules:

  → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent events

The probabilistic query evaluation problem ($\mathrm{PQE}(q)$)

Definition: problem $\mathrm{PQE}(q)$, for q a Boolean query

Input: a tuple-independent probabilistic database D

Output: $\Pr(D \models q)$

- Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the (data) complexity of $\mathrm{PQE}(q)$ for unions of conjunctive queries:
  - either $\mathrm{PQE}(q) \in \mathrm{PTIME}$, and q is called "safe"
  - or $\mathrm{PQE}(q)$ is $\mathrm{FP}^{\#\mathrm{P}}$**-hard**, and q is called "unsafe"
- Their algorithm for a safe query q essentially uses three rules:
  - → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent events
  - → Negation: $\Pr(\neg A) = 1 - \Pr(A)$

The probabilistic query evaluation problem ($\mathrm{PQE}(q)$)

Definition: problem $\mathrm{PQE}(q)$, for q a Boolean query

Input: a tuple-independent probabilistic database D

Output: $\Pr(D \models q)$

- Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the (data) complexity of $\mathrm{PQE}(q)$ for unions of conjunctive queries:
  - either $\mathrm{PQE}(q) \in \mathrm{PTIME}$, and q is called "safe"
  - or $\mathrm{PQE}(q)$ is $\mathrm{FP}^{\#\mathrm{P}}$**-hard**, and q is called "unsafe"
- Their algorithm for a safe query q essentially uses three rules:
  - → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent events
  - → Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - → Inclusion–exclusion: $\Pr(A \vee B \vee C \vee \ldots) = \Pr(A) + \Pr(B) + \ldots - \Pr(A \wedge B) - \Pr(A \wedge C) - \ldots + \Pr(A \wedge B \wedge C) + \ldots$

Provenance

Definition

The provenance $\text{Prov}(q, I)$ of query q on database D is the Boolean function with facts of D as variables and such that for every valuation $\tau : D \to \{0, 1\}$, $\text{Prov}(q, D)$ evaluates to TRUE under τ if and only if $\{f \in D | \tau(f) = 1\} \models q$

Provenance

Definition

The provenance $\text{Prov}(q, I)$ of query q on database D is the Boolean function with facts of D as variables and such that for every valuation $\tau : D \to \{0, 1\}$, $\text{Prov}(q, D)$ evaluates to TRUE under τ if and only if $\{f \in D | \tau(f) = 1\} \models q$

Possible representations:

- Boolean formulas
- Binary Decision Diagrams (OBDDs, FBDDs, etc)
- Boolean circuits

Provenance: example

$D =$

Likes		p
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$$q = \exists x, y, z \; : L(x, z) \land L(y, z) \land x \neq y$$

Provenance: example

$D =$

Likes		p
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$$\mathrm{Prov}(q, D) = [L(A, B) \wedge L(B, B)] \vee [L(A, B) \wedge L(J, B)] \vee [L(B, B) \wedge L(J, B)]$$

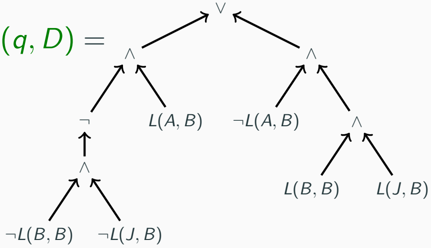
$$q = \exists x, y, z : L(x, z) \wedge L(y, z) \wedge x \neq y$$

Provenance: example

$D =$

Likes		**p**
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$\mathrm{Prov}(q, D) =$



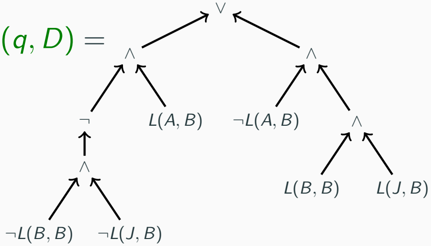
$q = \exists x, y, z \; : L(x, z) \wedge L(y, z) \wedge x \neq y$

Provenance: example

$D =$

Likes		p
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7

$\mathrm{Prov}(q, D) =$



$q = \exists x, y, z \; : L(x, z) \wedge L(y, z) \wedge x \neq y$

We have $\Pr(D \models q) = \Pr(\mathrm{Prov}(q, D) = \text{true})$

Provenance in knowledge compilation formalisms

$$\Pr(D \models q) = \Pr(\mathrm{Prov}(q, D) = \mathrm{true})$$

$\rightarrow$ If we can, in PTIME, compute $\mathrm{Prov}(q, D)$ in a formalism from knowledge compilation that allows PTIME probability computation, we can solve $\mathrm{PQE}(q)$ in PTIME

Provenance in knowledge compilation formalisms

$$\Pr(D \models q) = \Pr(\mathrm{Prov}(q, D) = \mathrm{true})$$

$\rightarrow$ If we can, in PTIME, compute $\mathrm{Prov}(q, D)$ in a formalism from knowledge compilation that allows PTIME probability computation, we can solve $\mathrm{PQE}(q)$ in PTIME

- free or ordered decision diagrams (OBDDs, FBDDs)

Provenance in knowledge compilation formalisms

$$\Pr(D \models q) = \Pr(\mathsf{Prov}(q, D) = \mathsf{true})$$

$\rightarrow$ If we can, in PTIME, compute $\mathsf{Prov}(q, D)$ in a formalism from knowledge compilation that allows PTIME probability computation, we can solve $\mathrm{PQE}(q)$ in PTIME

- free or ordered decision diagrams (OBDDs, FBDDs)
- deterministic and decomposable Boolean circuits (d-Ds)

Provenance in knowledge compilation formalisms

$$\Pr(D \models q) = \Pr(\mathrm{Prov}(q, D) = \mathrm{true})$$

$\rightarrow$ If we can, in PTIME, compute $\mathrm{Prov}(q, D)$ in a formalism from knowledge compilation that allows PTIME probability computation, we can solve $\mathrm{PQE}(q)$ in PTIME

- free or ordered decision diagrams (OBDDs, FBDDs)
- deterministic and decomposable Boolean circuits (d-Ds)

- Dan Suciu's talk: the safe UCQs for which this is possible with OBDDs are exactly the inversion-free UCQs

$\rightarrow$ This talk: what about d-Ds?

What are deterministic and decomposable circuits (d-Ds)?

Let C be a Boolean circuit

- a $\wedge$-gate g is decomposable if any two inputs gates g_1, g_2 of g depend on disjoint sets of variables

What are deterministic and decomposable circuits (d-Ds)?

Let C be a Boolean circuit

- a $\wedge$-gate g is decomposable if any two inputs gates g_1, g_2 of g depend on disjoint sets of variables
- a $\vee$-gate g is deterministic if any two inputs gates g_1, g_2 of g are mutually exclusive

What are deterministic and decomposable circuits (d-Ds)?

Let C be a Boolean circuit

- a $\wedge$-gate g is decomposable if any two inputs gates g_1, g_2 of g depend on disjoint sets of variables
- a $\vee$-gate g is deterministic if any two inputs gates g_1, g_2 of g are mutually exclusive
- the circuit C is a d-D if all its $\wedge$-gates are decomposable and all its $\vee$-gates are deterministic

What are deterministic and decomposable circuits (d-Ds)?

Let C be a Boolean circuit

- a $\wedge$-gate g is decomposable if any two inputs gates g_1, g_2 of g depend on disjoint sets of variables
- a $\vee$-gate g is deterministic if any two inputs gates g_1, g_2 of g are mutually exclusive
- the circuit C is a d-D if all its $\wedge$-gates are decomposable and all its $\vee$-gates are deterministic

→ To obtain the probability, replace $\wedge$-gates by $\times$, $\vee$-gates by $+$, $\neg$-gates by $1 - x$, and evaluate. In other words, use the following rules:

  → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent events

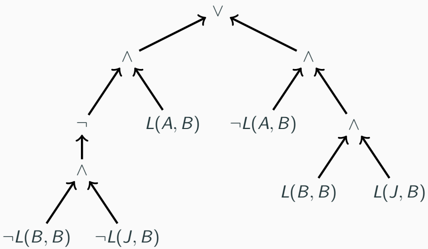
  → Negation: $\Pr(\neg A) = 1 - \Pr(A)$

  → Disjoint Events: $\Pr(A \vee B) = \Pr(A) + \Pr(B)$ for A, B disjoint events

d-Ds: example

$D =$

Likes		p
Alice	Bob	0.5
Alice	John	1
Bob	Bob	0.2
John	Bob	0.7



$q = \exists x, y, z\ :\ L(x, z) \wedge L(y, z) \wedge x \neq y$

The Intensional-Extensional problem

Intensional-Extensional (open) problem for d-Ds

For every safe UCQ q, can we compute in PTIME its provenance on a database D as a deterministic and decomposable circuit?

The Intensional-Extensional problem

Intensional-Extensional (open) problem for d-Ds

For every safe UCQ q, can we compute in PTIME its provenance on a database D as a deterministic and decomposable circuit?

In other words, can we replace the inclusion–exclusion rule by the disjunction rule?

The Intensional-Extensional problem

Intensional-Extensional (open) problem for d-Ds

For every safe UCQ q, can we compute in PTIME its provenance on a database D as a deterministic and decomposable circuit?

In other words, can we replace the inclusion–exclusion rule by the disjunction rule?

→ This approach is more modular than Dalvi and Suciu's original algorithm for safe UCQs, and it would allow us to do more than probabilistic evaluation: enumerate the satisfying states of the data, compute the satisfying state of the data that is most probable, update the tuples' probabilities, etc.

Solving the problem for a specific class of UCQs

Main result from PODS'20

- Focus on a class of UCQs, denoted $\mathcal{H}$ (defined next slide)
- It had been conjectured that for some safe queries $q \in \mathcal{H}$, the provenance of q cannot be computed in PTIME as d-Ds
  - $\rightarrow$ because these are the simplest queries for which Dalvi and Suciu's algorithm uses **inclusion–exclusion**
  - $\rightarrow$ because this conjecture had been proven for more restricted formalisms of knowledge compilation (d-SDNNFs, dec-DNNFs)

Main result

For every (fixed) safe query $q \in \mathcal{H}$, being given as input a database D, we can compute in PTIME a d-D that represents $\mathrm{Prov}(q, D)$.

The $\mathcal{H}$ queries

- Let $k \geq 1$ and $R, S_1, \ldots, S_k, T$ be pairwise distinct relational predicates, with R and T unary and S_i binary. Define the queries $h_{k,i}$ for $0 \leq i \leq k$:

The $\mathcal{H}$ queries

- Let $k \geq 1$ and $R, S_1, \ldots, S_k, T$ be pairwise distinct relational predicates, with R and T unary and S_i binary. Define the queries $h_{k,i}$ for $0 \leq i \leq k$:
  - $h_{k,0} \stackrel{\text{def}}{=} \exists x \exists y \; R(x) \land S_1(x, y)$;
  - $h_{k,i} \stackrel{\text{def}}{=} \exists x \exists y \; S_i(x, y) \land S_{i+1}(x, y)$ for $1 \leq i < k$;
  - $h_{k,k} \stackrel{\text{def}}{=} \exists x \exists y \; S_k(x, y) \land T(y)$.

The $\mathcal{H}$ queries

- Let $k \geq 1$ and $R, S_1, \ldots, S_k, T$ be pairwise distinct relational predicates, with R and T unary and S_i binary. Define the queries $h_{k,i}$ for $0 \leq i \leq k$:
  - $h_{k,0} \stackrel{\text{def}}{=} \exists x \exists y \; R(x) \wedge S_1(x, y)$;
  - $h_{k,i} \stackrel{\text{def}}{=} \exists x \exists y \; S_i(x, y) \wedge S_{i+1}(x, y)$ for $1 \leq i < k$;
  - $h_{k,k} \stackrel{\text{def}}{=} \exists x \exists y \; S_k(x, y) \wedge T(y)$.
- $\mathcal{H}_k \stackrel{\text{def}}{=}$ the set of UCQs that can be formed from the queries $h_{k,i}$, i.e., positive Boolean combinations of those queries
- $\mathcal{H} \stackrel{\text{def}}{=} \bigcup_{k=1}^{\infty} \mathcal{H}_k$

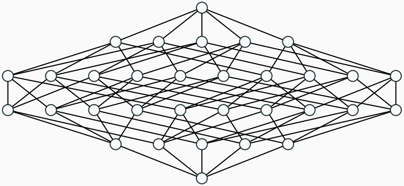
Proof technique (1/4): representing $\mathcal{H}$ queries

Write $[k] \stackrel{\text{def}}{=} \{0, \ldots, k\}$. Let us represent a query $q \in \mathcal{H}_k$ as follows:

Proof technique (1/4): representing $\mathcal{H}$ queries

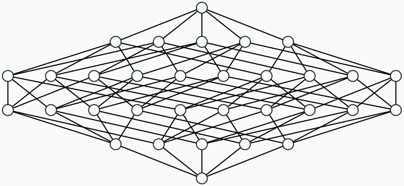
Write $[k] \stackrel{\text{def}}{=} \{0, \ldots, k\}$. Let us represent a query $q \in \mathcal{H}_k$ as follows:

- the (Hasse diagram of) Boolean lattice of $2^{[k]}$



Proof technique (1/4): representing $\mathcal{H}$ queries

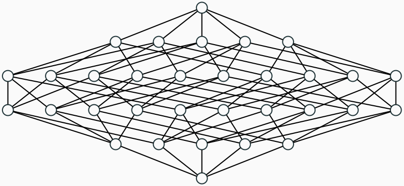
Write $[k] \stackrel{\text{def}}{=} \{0, \ldots, k\}$. Let us represent a query $q \in \mathcal{H}_k$ as follows:



- the (Hasse diagram of) Boolean lattice of $2^{[k]}$
- each node $v \subseteq [k]$ of the graph represents a subquery $q_v \stackrel{\text{def}}{=} \left(\bigwedge_{\ell \in v} h_{k,\ell}\right) \wedge \left(\bigwedge_{\ell \in [k] \setminus v} \neg h_{k,\ell}\right)$. (Note that q_v is not a UCQ)

Proof technique (1/4): representing $\mathcal{H}$ queries

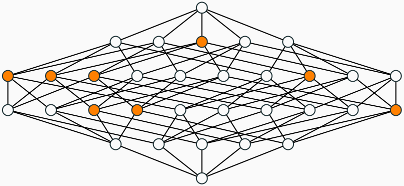
Write $[k] \stackrel{\text{def}}{=} \{0, \ldots, k\}$. Let us represent a query $q \in \mathcal{H}_k$ as follows:



- the (Hasse diagram of) Boolean lattice of $2^{[k]}$
- each node $v \subseteq [k]$ of the graph represents a subquery $q_v \stackrel{\text{def}}{=} \left(\bigwedge_{\ell \in v} h_{k,\ell}\right) \wedge \left(\bigwedge_{\ell \in [k] \setminus v} \neg h_{k,\ell}\right)$. (Note that q_v is not a UCQ)
- (in particular, every database D satisfies exactly one subquery q_v)

Proof technique (1/4): representing $\mathcal{H}$ queries

Write $[k] \stackrel{\text{def}}{=} \{0, \ldots, k\}$. Let us represent a query $q \in \mathcal{H}_k$ as follows:

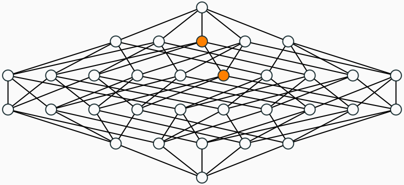


- the (Hasse diagram of) Boolean lattice of $2^{[k]}$
- each node $v \subseteq [k]$ of the graph represents a subquery $q_v \stackrel{\text{def}}{=} \left(\bigwedge_{\ell \in v} h_{k,\ell}\right) \land \left(\bigwedge_{\ell \in [k] \setminus v} \neg h_{k,\ell}\right)$. (Note that q_v is not a UCQ)
- (in particular, every database D satisfies exactly one subquery q_v)
- some nodes are colored, and $q =$ the disjunction of the subqueries q_v that are represented by the colored nodes v

Proof technique (2/4): basic queries

Proposition (Fink & Olteanu [TODS'16])

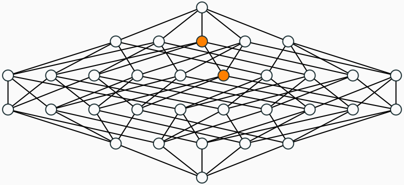
For any adjacent nodes v, v' of the graph, being given as input a database D, we can compute in PTIME a d-D representing $\mathrm{Prov}(q_v \vee q_{v'}, D)$.



Proof technique (2/4): basic queries

Proposition (Fink & Olteanu [TODS'16])

For any adjacent nodes v, v' of the graph, being given as input a database D, we can compute in PTIME a d-D representing $\mathrm{Prov}(q_v \vee q_{v'}, D)$.

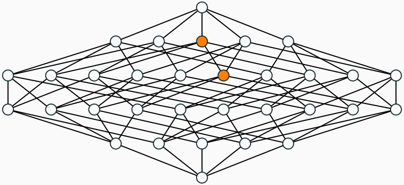


- **Idea**: starting from q, we will entirely uncolor the graph by using multiple times the following operations:
  - Uncolor two adjacent nodes that are colored
  - Color two adjacent nodes that were not colored

Proof technique (2/4): basic queries

Proposition (Fink & Olteanu [TODS'16])

For any adjacent nodes v, v' of the graph, being given as input a database D, we can compute in PTIME a d-D representing $\mathrm{Prov}(q_v \vee q_{v'}, D)$.

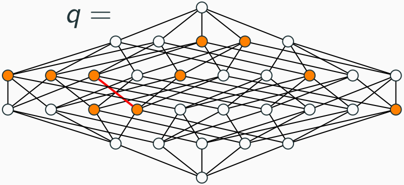


- **Idea**: starting from q, we will entirely uncolor the graph by using multiple times the following operations:
  - Uncolor two adjacent nodes that are colored
  - Color two adjacent nodes that were not colored

→ Simultaneously, we build a deterministic and decomposable circuit for the provenance of q

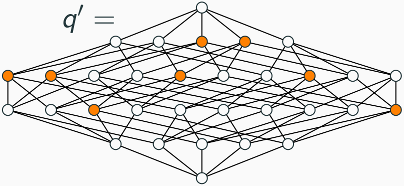
Uncoloring:

$q =$



$\downarrow$

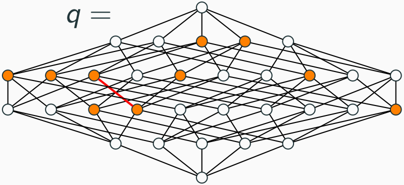
$q' =$



Proof technique (3/4): circuit construction

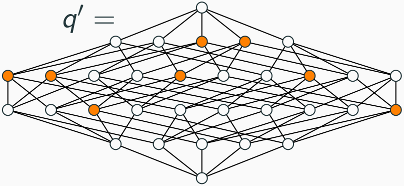
Uncoloring:

$q =$

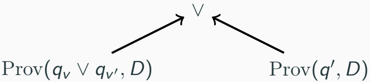


$\downarrow$

$q' =$



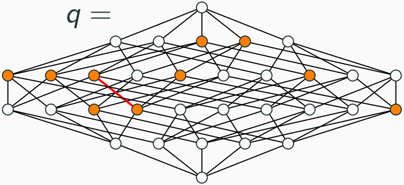
$$\mathrm{Prov}(q, D) =$$



Proof technique (3/4): circuit construction

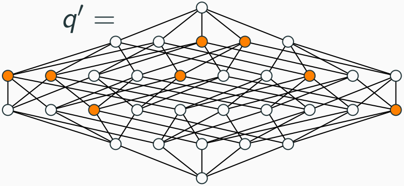
Uncoloring:

$q =$

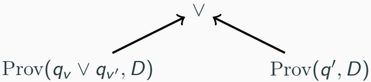


$\downarrow$

$q' =$



$\mathrm{Prov}(q, D) =$

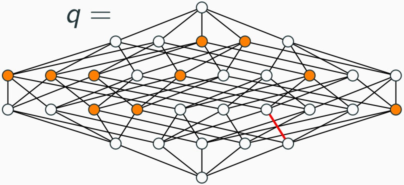


Then continue with q'

Proof technique (3/4): circuit construction

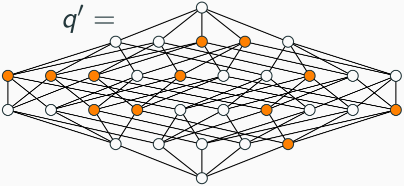
Coloring: (Guy Van den Broeck's trick)

$q =$



↓

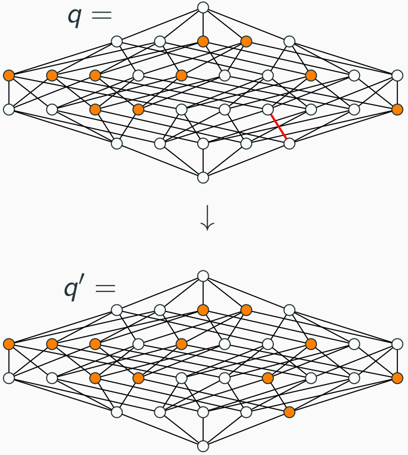
$q' =$



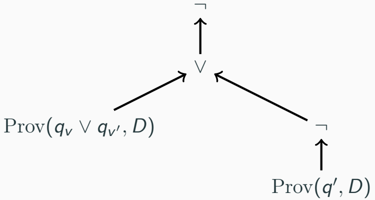
$\mathrm{Prov}(q, D) =$

Proof technique (3/4): circuit construction

Coloring: (Guy Van den Broeck's trick)



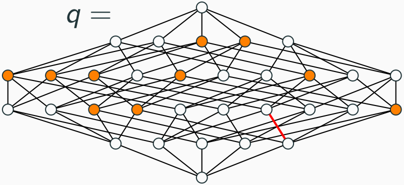
$\mathrm{Prov}(q, D) =$



Proof technique (3/4): circuit construction

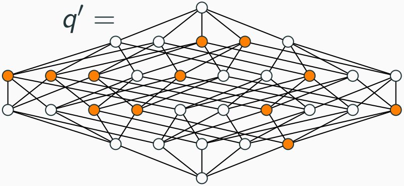
Coloring: (Guy Van den Broeck's trick)

$q =$

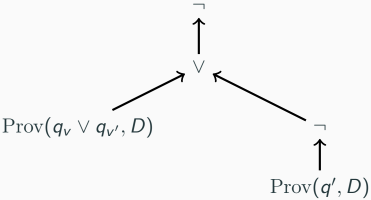


$\downarrow$

$q' =$

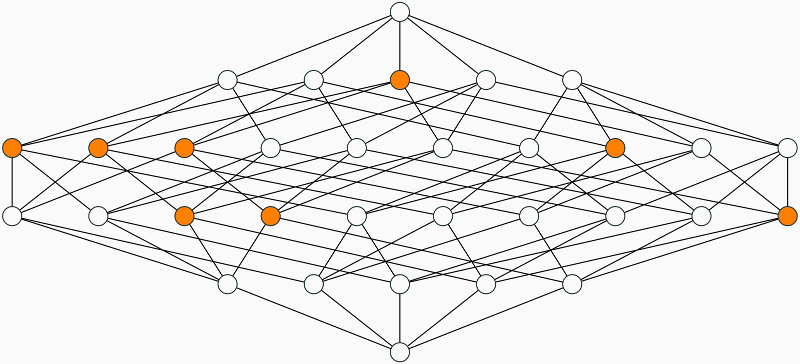


$\mathrm{Prov}(q, D) =$

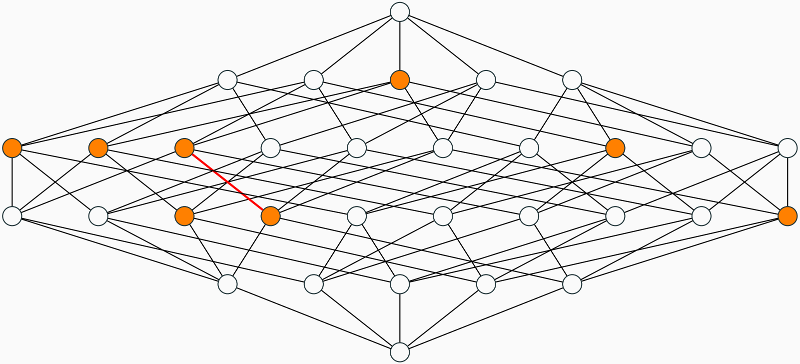


Then continue with q'

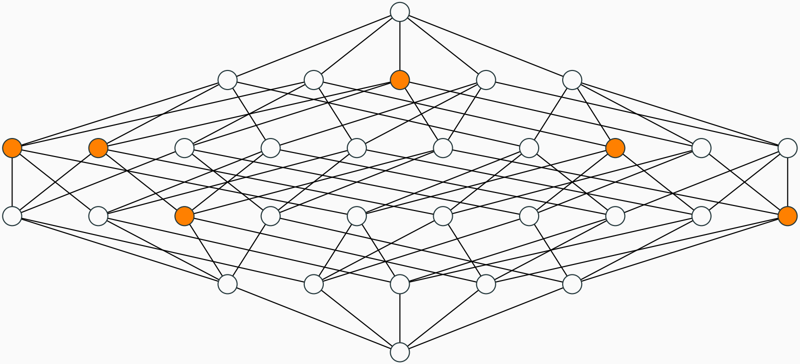
Proof technique (4/4): how to uncolor the graph?



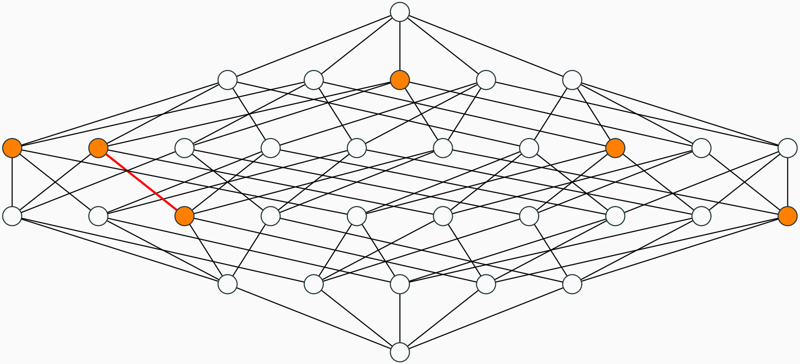
Proof technique (4/4): how to uncolor the graph?



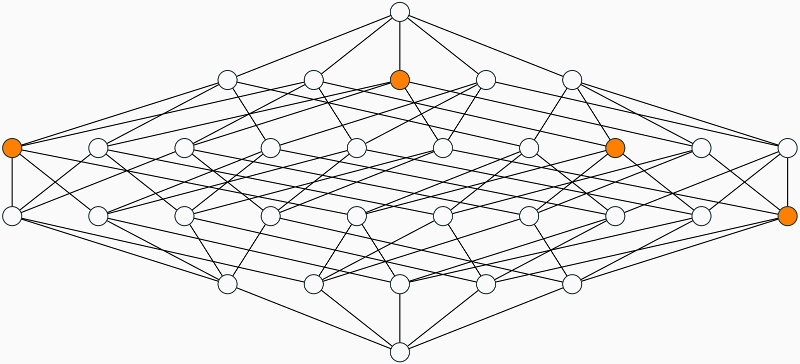
Proof technique (4/4): how to uncolor the graph?



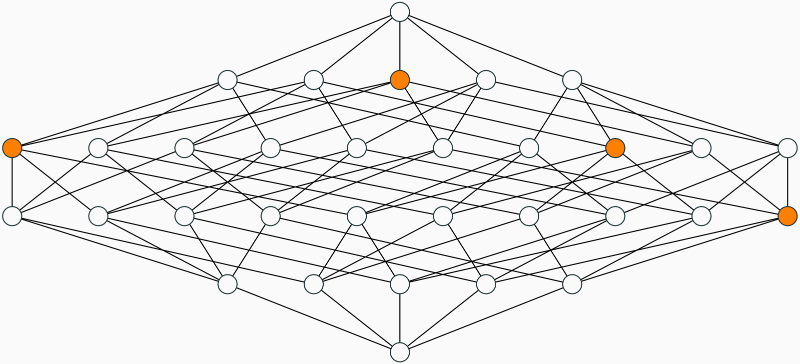
Proof technique (4/4): how to uncolor the graph?



Proof technique (4/4): how to uncolor the graph?



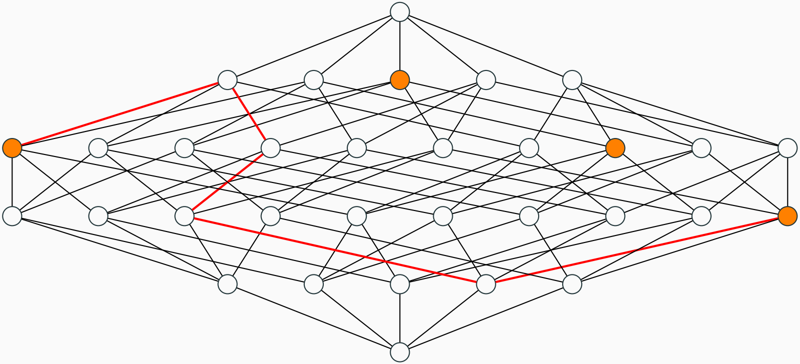
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

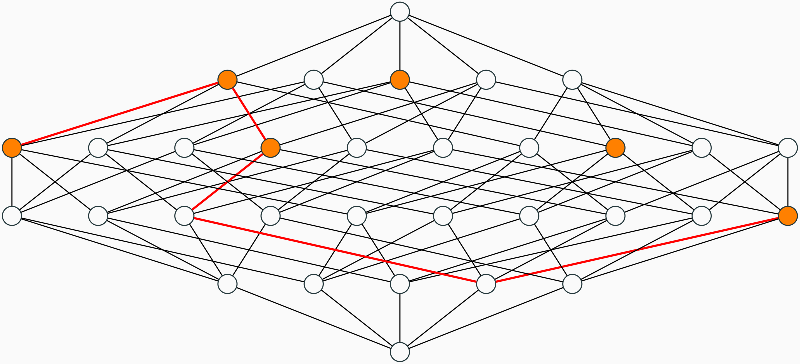
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

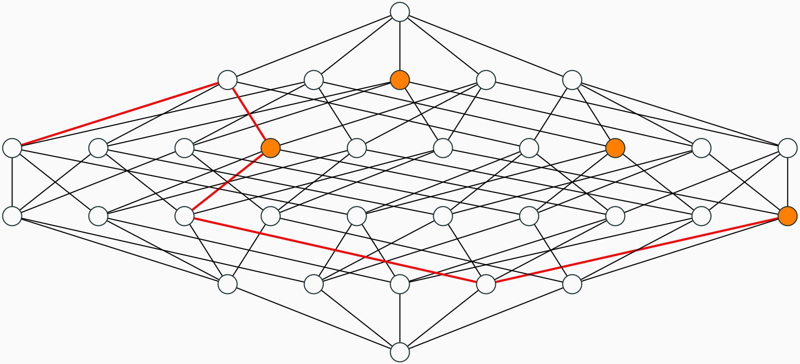
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

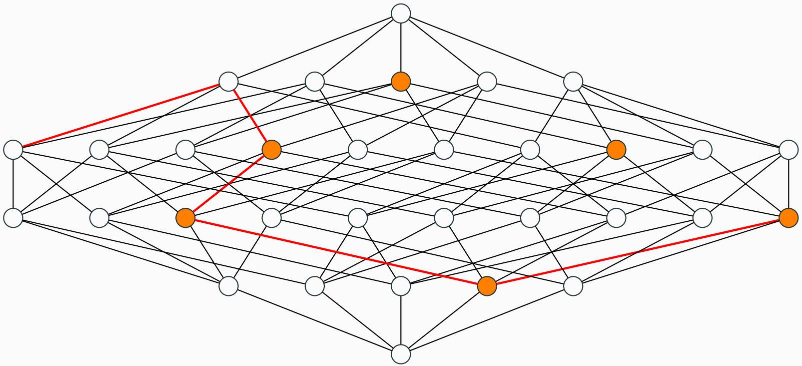
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

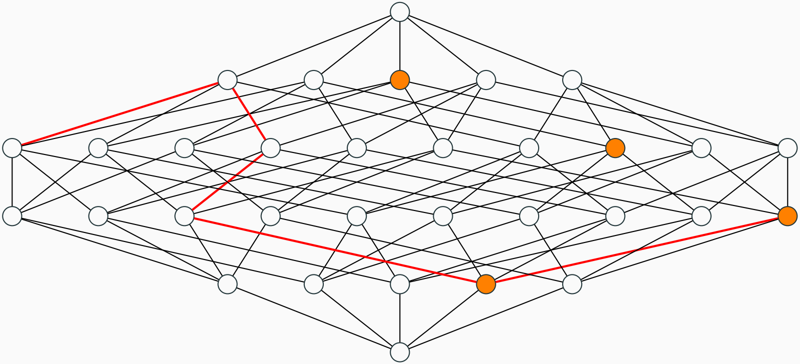
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

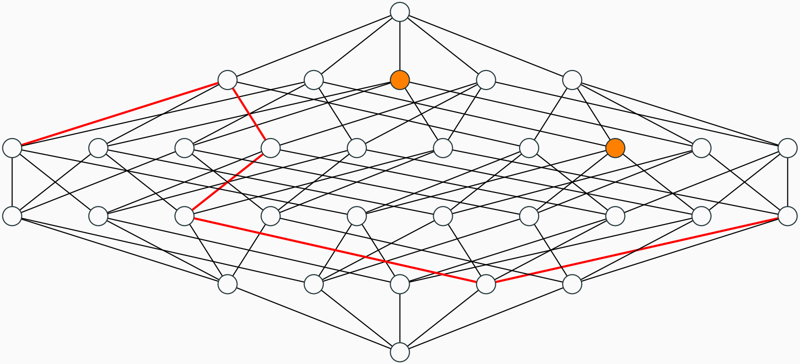
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

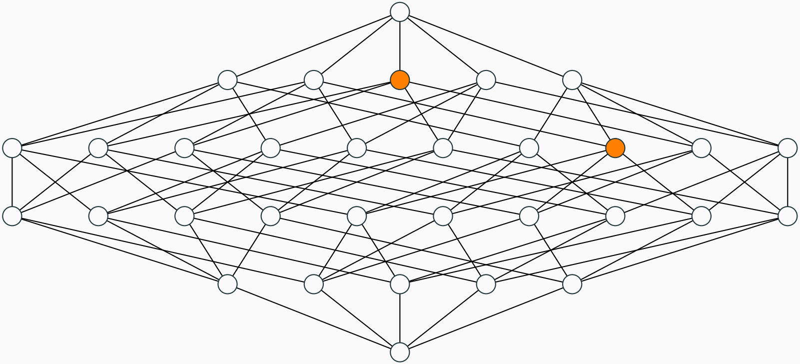
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

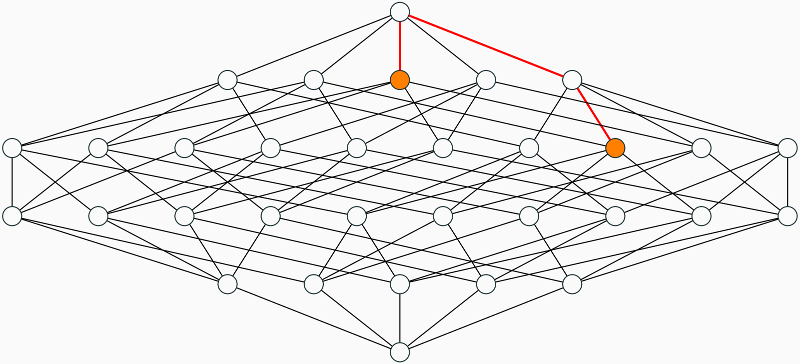
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

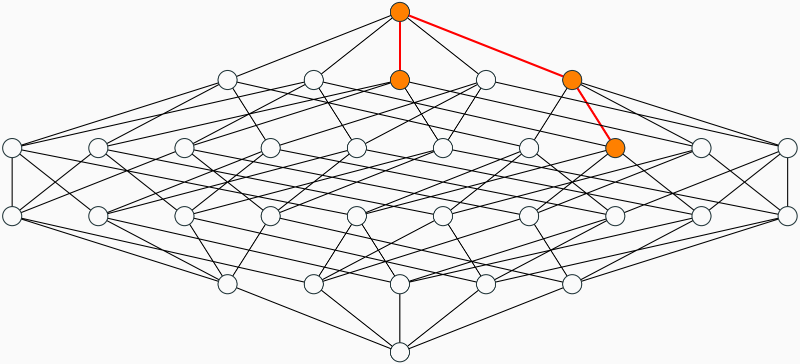
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

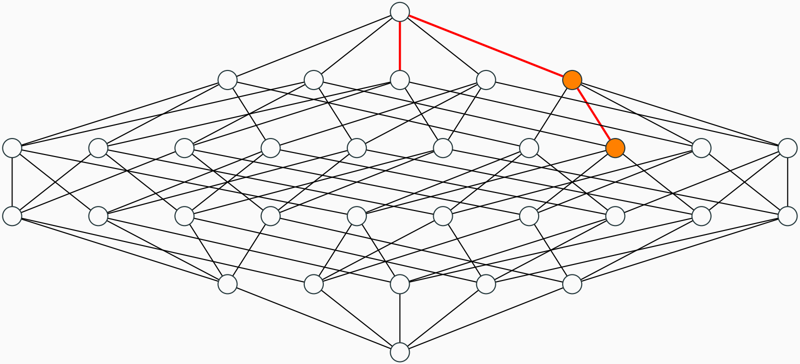
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

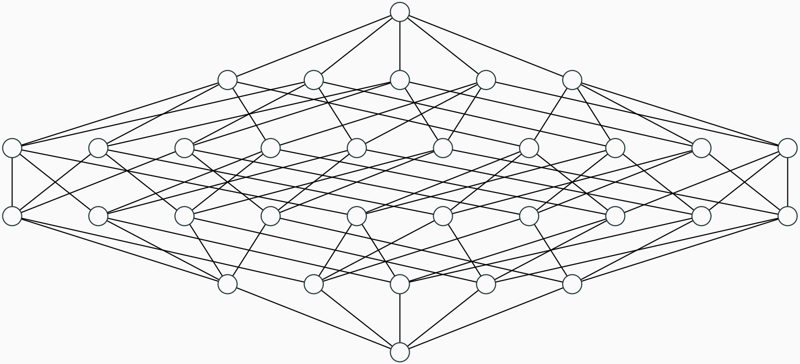
Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

Proof technique (4/4): how to uncolor the graph?



Proposition

A query $q \in \mathcal{H}_k$ is safe if and only if the two partitions of the graph contain the same number of colored nodes

The non-cancelling intersections conjecture

Co-workers

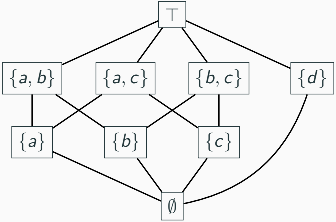
Ongoing work with Antoine Amarilli, Louis Jachiet and Dan Suciu

Intersection lattices, Möbius function and Inclusion-Exclusion

- Let $\mathcal{F} = \{S_1, \dots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b, c\}, \{d\}\}$

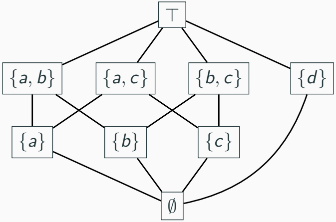
Intersection lattices, Möbius function and Inclusion-Exclusion

- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b, c\}, \{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



Intersection lattices, Möbius function and Inclusion-Exclusion

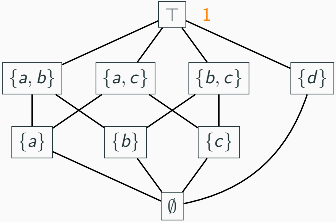
- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b, c\}, \{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



- Let $\mu_{\mathcal{F}} : \mathbb{L}_{\mathcal{F}} \rightarrow \mathbb{Z}$ be the Möbius function defined by

Intersection lattices, Möbius function and Inclusion-Exclusion

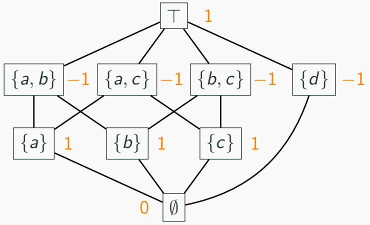
- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a,b\},\{a,c\},\{b,c\},\{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



- Let $\mu_{\mathcal{F}} : \mathbb{L}_{\mathcal{F}} \rightarrow \mathbb{Z}$ be the Möbius function defined by
  - $\mu_{\mathcal{F}}(\top) = 1$

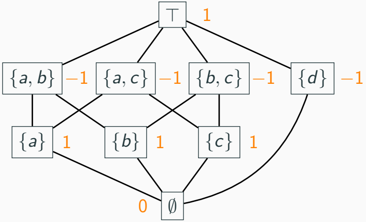
Intersection lattices, Möbius function and Inclusion-Exclusion

- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a,b\},\{a,c\},\{b,c\},\{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



- Let $\mu_{\mathcal{F}} : \mathbb{L}_{\mathcal{F}} \rightarrow \mathbb{Z}$ be the Möbius function defined by
  - $\mu_{\mathcal{F}}(\top) = 1$
  - $\mu_{\mathcal{F}}(I) = -\sum_{\substack{I' \in \mathbb{L}_{\mathcal{F}} \\ I' > I}} \mu_{\mathcal{F}}(I')$ for $I \in \mathbb{L}_{\mathcal{F}}$, $I \neq \top$

- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  - $\rightarrow$ **Example:** $\mathcal{F} = \{\{a,b\},\{a,c\},\{b,c\},\{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



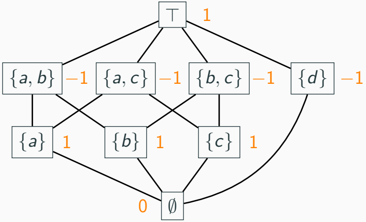
- Let $\mu_{\mathcal{F}} : \mathbb{L}_{\mathcal{F}} \rightarrow \mathbb{Z}$ be the Möbius function defined by
  - $\mu_{\mathcal{F}}(\top) = 1$
  - $\mu_{\mathcal{F}}(I) = -\sum_{\substack{I' \in \mathbb{L}_{\mathcal{F}} \\ I' > I}} \mu_{\mathcal{F}}(I')$ for $I \in \mathbb{L}_{\mathcal{F}}$, $I \neq \top$

Fact (coefficients of the Inclusion-Exclusion formula)

$$|\bigcup_{i=1}^{n} S_i| = -\sum_{\substack{I \in \mathbb{L}_{\mathcal{F}} \\ I \neq \top}} \mu_{\mathcal{F}}(I) \times |I|$$

Intersection lattices, Möbius function and Inclusion-Exclusion

- Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets, pairwise incomparable
  → **Example:** $\mathcal{F} = \{\{a,b\},\{a,c\},\{b,c\},\{d\}\}$
- Let $\mathbb{L}_{\mathcal{F}}$ be its intersection lattice:



- Let $\mu_{\mathcal{F}} : \mathbb{L}_{\mathcal{F}} \to \mathbb{Z}$ be the Möbius function defined by
  - $\mu_{\mathcal{F}}(\top) = 1$
  - $\mu_{\mathcal{F}}(I) = -\sum_{\substack{I' \in \mathbb{L}_{\mathcal{F}} \\ I' > I}} \mu_{\mathcal{F}}(I')$ for $I \in \mathbb{L}_{\mathcal{F}}$, $I \neq \top$

Fact (coefficients of the Inclusion-Exclusion formula)

$$|\bigcup_{i=1}^{n} S_i| = -\sum_{\substack{I \in \mathbb{L}_{\mathcal{F}} \\ I \neq \top}} \mu_{\mathcal{F}}(I) \times |I|$$

- Define the non-cancelling intersections of $\mathcal{F}$ by
  $\mathtt{NCI}(\mathcal{F}) \stackrel{\text{def}}{=} \{I \in \mathbb{L}_{\mathcal{F}} \mid I \neq \top \text{ and } \mu_{\mathcal{F}}(I) \neq 0\}$

Non-cancelling intersections conjecture

- For two sets S, T such that $S \cap T = \emptyset$, define the disjoint union $S \mathbin{\dot{\cup}} T \stackrel{\text{def}}{=} S \cup T$
- For two sets S, T such that $T \subseteq S$, define the subset complement $S \mathbin{\dot{\setminus}} T \stackrel{\text{def}}{=} S \setminus T$

Non-cancelling intersections conjecture

- For two sets S, T such that $S \cap T = \emptyset$, define the disjoint union $S \mathbin{\dot\cup} T \stackrel{\text{def}}{=} S \cup T$
- For two sets S, T such that $T \subseteq S$, define the subset complement $S \mathbin{\dot\setminus} T \stackrel{\text{def}}{=} S \setminus T$

- For a set family $\mathcal{T}$, define $\bullet(\mathcal{T})$ to be the smallest set family which contains all the sets of $\mathcal{T}$ and is closed under disjoint union and subset complement

Non-cancelling intersections conjecture

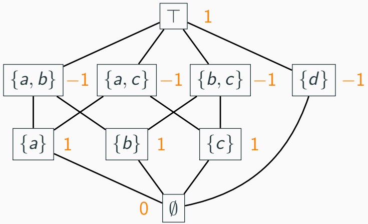
- For two sets S, T such that $S \cap T = \emptyset$, define the disjoint union $S \mathbin{\dot\cup} T \stackrel{\text{def}}{=} S \cup T$
- For two sets S, T such that $T \subseteq S$, define the subset complement $S \mathbin{\dot\setminus} T \stackrel{\text{def}}{=} S \setminus T$

- For a set family $\mathcal{T}$, define $\bullet(\mathcal{T})$ to be the smallest set family which contains all the sets of $\mathcal{T}$ and is closed under disjoint union and subset complement

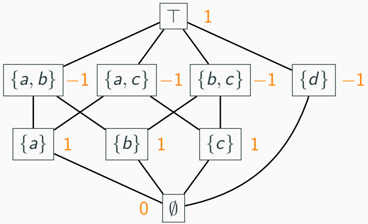
Non-cancelling intersections conjecture (NCI for short)

Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ be a finite family of finite sets.
Then $\bigcup_{i=1}^{n} S_i \in \bullet(\mathtt{NCI}(\mathcal{F}))$.

Example 1

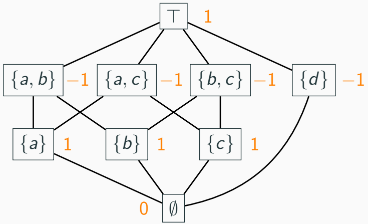


Example 1



→ We have $\bigcup_{i=1}^{n} S_i = \{a, b, c, d\} = ((\{a\} \dot{\cup} \{b\}) \dot{\cup} \{c\}) \dot{\cup} \{d\}$

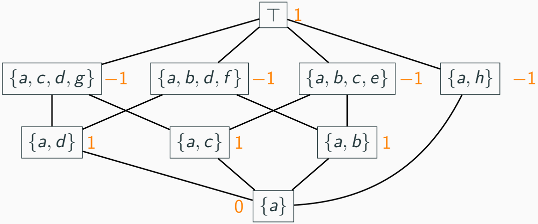
Example 1



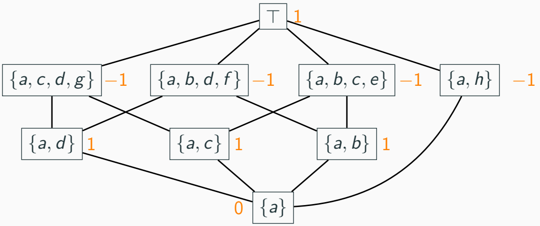
→ We have $\bigcup_{i=1}^{n} S_i = \{a, b, c, d\} = ((\{a\} \dot\cup \{b\}) \dot\cup \{c\}) \dot\cup \{d\}$

That was easy...

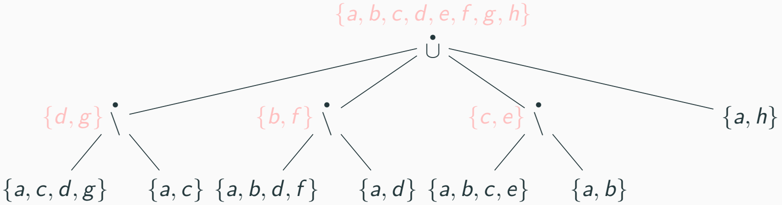
Example 2



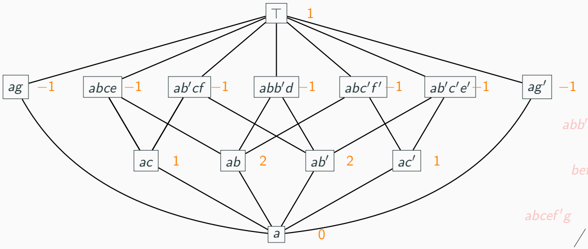
Example 2



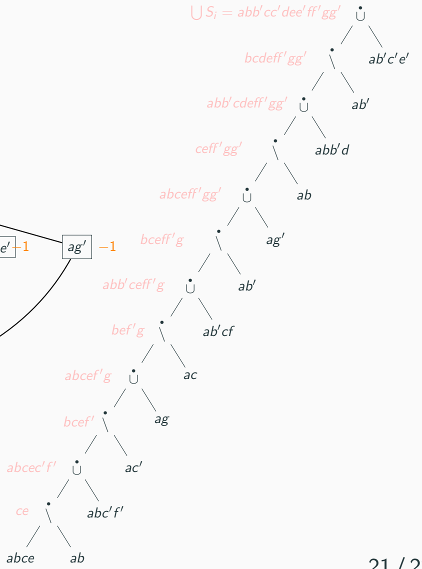
→ We can express $\bigcup_{i=1}^{n} S_i = \{a, b, c, d, e, f, g, h\}$ with:



Example 3



⊤
1
ag
−1
abce
−1
ab′cf
−1
abb′d
−1
abc′f′
−1
ab′c′e′
−1
ag′
−1
ac
1
ab
2
ab′
2
ac′
1
a
0



$\bigcup S_i = abb'cc'dee'ff'gg'$
ab′c′e′
bcdeff′gg′
ab′
abb′cdeff′gg′
abb′d
ceff′gg′
ab
abceff′gg′
ag′
bceff′g
ab′
abb′ceff′g
ab′cf
bef′g
ac
abcef′g
ag
bcef′
ac′
abcec′f′
abc′f′
ce
abce
ab

Conclusion

- We have sketched a proof that we can build in PTIME d-Ds for the provenance of safe queries in the class $\mathcal{H}$
- We have stated a more general conjecture about intersection lattices: the non-cancelling intersections conjecture

Conclusion

- We have sketched a proof that we can build in PTIME d-Ds for the provenance of safe queries in the class $\mathcal{H}$
- We have stated a more general conjecture about intersection lattices: the non-cancelling intersections conjecture
  - → Counterexample search by bruteforce: no counterexample so far...

Conclusion

- We have sketched a proof that we can build in PTIME d-Ds for the provenance of safe queries in the class $\mathcal{H}$

- We have stated a more general conjecture about intersection lattices: the non-cancelling intersections conjecture
  - → Counterexample search by bruteforce: no counterexample so far...
  - → We have some partial positive results: a reformulation of the conjecture that works in the Boolean lattices, and a proof for specific subcases of this reformulation

Thanks for your attention!

Nilesh N. Dalvi and Dan Suciu.
The dichotomy of probabilistic inference for unions of conjunctive queries.
Journal of the ACM, 59(6):30, 2012.

Robert Fink and Dan Olteanu.
Dichotomies for queries with negation in probabilistic databases.
ACM Transactions on Database Systems (TODS), 41(1):4, 2016.

Bibliography ii

Mikaël Monet.
Solving a special case of the intensional vs extensional conjecture in probabilistic databases.
In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 149–163, 2020.