Solving a Special Case of the Intensional vs Extensional Conjecture in Probabilistic Databases

Mikaël Monet

June 15th, 2020

Millennium Institute
Foundational
Research on Data

Probabilistic databases by example

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

Probabilistic databases by example

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

D =

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

Probabilistic databases by example

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

$D' =$

Likes		π
		0.5
Alice	John	1
		0.2
John	Bob	0.7

Probabilistic databases by example

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

$D' =$

Likes		π
		0.5
Alice	John	1
		0.2
John	Bob	0.7

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

Probabilistic databases by example

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

D =

Likes		π
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"

Probabilistic databases by example

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

$D =$

Likes		π
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"

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

Probabilistic databases by example

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

D =

Likes		π
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"

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

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

Probabilistic databases by example

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

D =

Likes		π
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"

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

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

Probabilistic databases by example

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

$D =$

Likes		π
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"

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

$$\Pr(D \vDash q) = 0.5 \times [1 - (1 - 0.2)(1 - 0.7)]$$

Probabilistic databases by example

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

$D =$

Likes		π
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"

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

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

$$+ (1 - 0.5) \times [0.2 \times 0.7]$$

The $\mathrm{PQE}(q)$ problem

Definition: problem $\mathrm{PQE}(q)$

Input: a probabilistic database D

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

→ Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the 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**

The $\mathrm{PQE}(q)$ problem

Definition: problem $\mathrm{PQE}(q)$

Input: a probabilistic database D

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

→ Dalvi and Suciu [JACM'12] have shown a **dichotomy** on the 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**

- However, their algorithm:
  - → can only be used for PQE
  - → does not make the connection with knowledge compilation

An open question

- Dalvi et Suciu's algorithm to solve $\mathrm{PQE}(q)$ for a safe query q essentially uses three rules:
  - → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent
  - → Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - → Inclusion–exclusion: $\Pr(A \vee B \vee C) = \Pr(A) + \Pr(B) + \Pr(C) - \Pr(A \wedge B) - \Pr(A \wedge C) - \Pr(B \wedge C) + \Pr(A \wedge B \wedge C)$

Open problem

Can we replace the inclusion–exclusion rule by the disjunction rule ($\Pr(A \vee B) = \Pr(A) + \Pr(B)$ for A, B disjoint)?

An open question

- Dalvi et Suciu's algorithm to solve $\mathrm{PQE}(q)$ for a safe query q essentially uses three rules:
  - → Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when A, B are independent
  - → Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - → Inclusion–exclusion: $\Pr(A \vee B \vee C) = \Pr(A) + \Pr(B) + \Pr(C) - \Pr(A \wedge B) - \Pr(A \wedge C) - \Pr(B \wedge C) + \Pr(A \wedge B \wedge C)$

Open problem

Can we replace the inclusion–exclusion rule by the disjunction rule ($\Pr(A \vee B) = \Pr(A) + \Pr(B)$ for A, B disjoint)?

→ **In other words**: for every safe query q, can we solve $\mathrm{PQE}(q)$ in polynomial time by using only the independence, negation, and disjunction rules?

Formalization: Knowledge compilation

→ For every safe query q, can we solve $\mathrm{PQE}(q)$ in PTIME by using only the independence, negation, and disjunction rules?

(The precise formalization of this problem uses Boolean circuit classes from knowledge compilation: "For every safe query q, can we compute in polynomial time its provenance on a database D as a deterministic and decomposable circuit?")

Formalization: Knowledge compilation

→ For every safe query q, can we solve $\mathrm{PQE}(q)$ in PTIME by using only the independence, negation, and disjunction rules?

(The precise formalization of this problem uses Boolean circuit classes from knowledge compilation: "For every safe query q, can we compute in polynomial time its provenance on a database D as a deterministic and decomposable circuit?")

Why this problem?

→ This 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.

→ Approach is more modular, and would justify the use of knowledge compilation for PQE

Contribution

- We focus on a query class, denoted $\mathcal{H}$
- It had been conjectured that for the safe queries of $\mathcal{H}$, $\mathrm{PQE}(q)$ cannot be solved with knowledge compilation
  - → because these are the simplest queries for which Dalvi and Suciu's algorithm uses **inclusion–exclusion**
  - → this conjecture has been proven for more restricted formalisms of knowledge compilation (d-SDNNFs, DLDDs)

Main result

For every safe query $q \in \mathcal{H}$, being given as input a database D, we can compute $\mathrm{Pr}(D \models q)$ in PTIME using only the negation, independence and disjunction rules.

Contribution

- We focus on a query class, denoted $\mathcal{H}$
- It had been conjectured that for the safe queries of $\mathcal{H}$, $\mathrm{PQE}(q)$ cannot be solved with knowledge compilation
  - → because these are the simplest queries for which Dalvi and Suciu's algorithm uses **inclusion–exclusion**
  - → this conjecture has been proven for more restricted formalisms of knowledge compilation (d-SDNNFs, DLDDs)

Main result

For every safe query $q \in \mathcal{H}$, being given as input a database D, we can compute $\mathrm{Pr}(D \models q)$ in PTIME using only the negation, independence and disjunction rules.

→ **No need of inclusion–exclusion!**

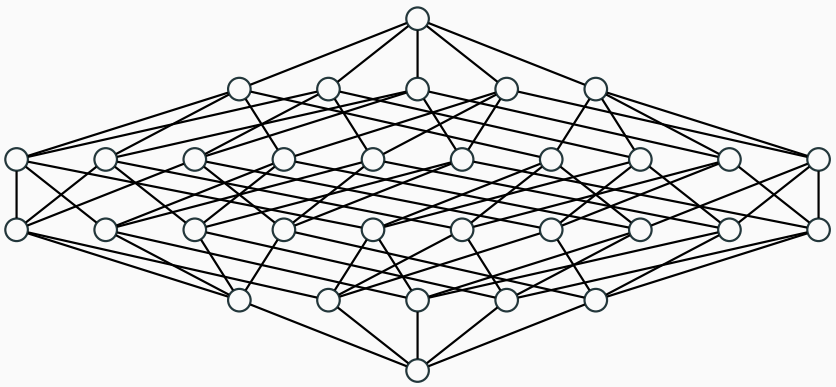
Proof technique (1/4): representing $\mathcal{H}$ queries

Let us represent a query $q \in \mathcal{H}$ as follows:

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

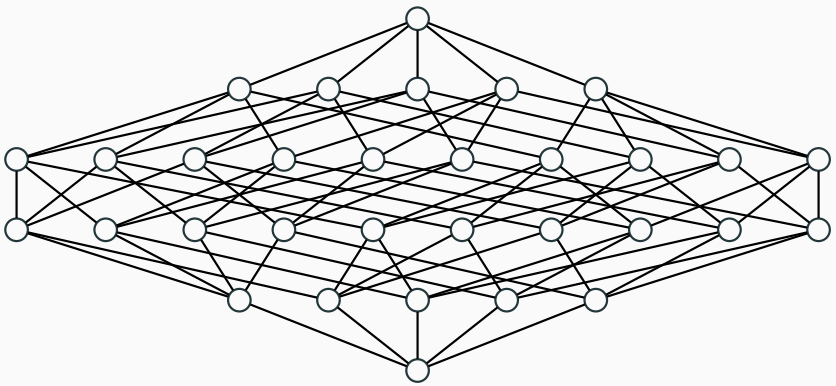
Let us represent a query $q \in \mathcal{H}$ as follows:

- the Boolean lattice (bipartite connected graph)



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

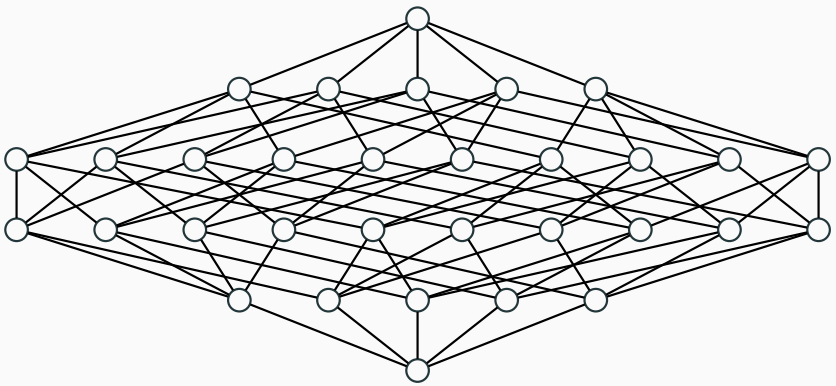
Let us represent a query $q \in \mathcal{H}$ as follows:



- the Boolean lattice (bipartite connected graph)
- each node v of the graph represents a (Boolean) subquery q_v

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

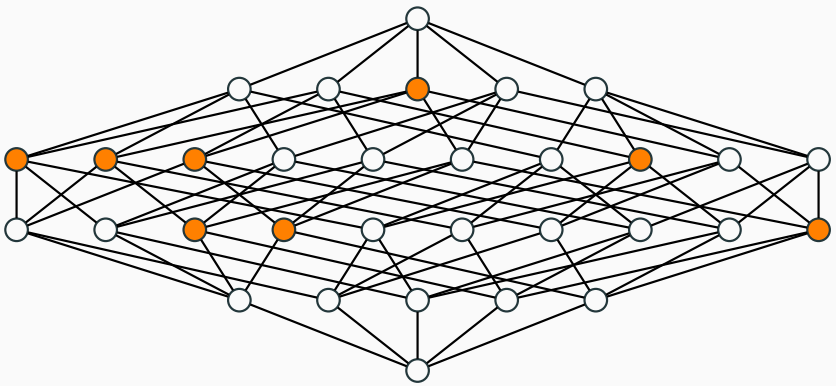
Let us represent a query $q \in \mathcal{H}$ as follows:



- the Boolean lattice (bipartite connected graph)
- each node v of the graph represents a (Boolean) subquery q_v
- every database D satisfies exactly one subquery q_v

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

Let us represent a query $q \in \mathcal{H}$ as follows:

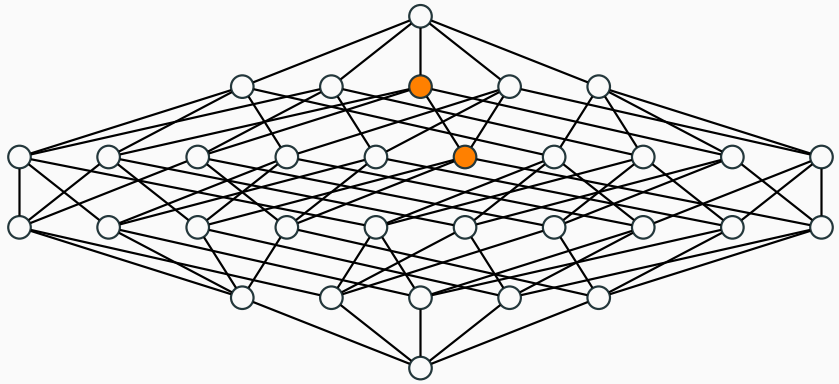


- the Boolean lattice (bipartite connected graph)
- each node v of the graph represents a (Boolean) subquery q_v
- every database D satisfies exactly one subquery q_v
- some nodes are colored, and q = the disjunction of the subqueries that are represented by the colored nodes

Proof technique (2/4): basic queries

Proposition (Fink & Olteanu [TODS'16])

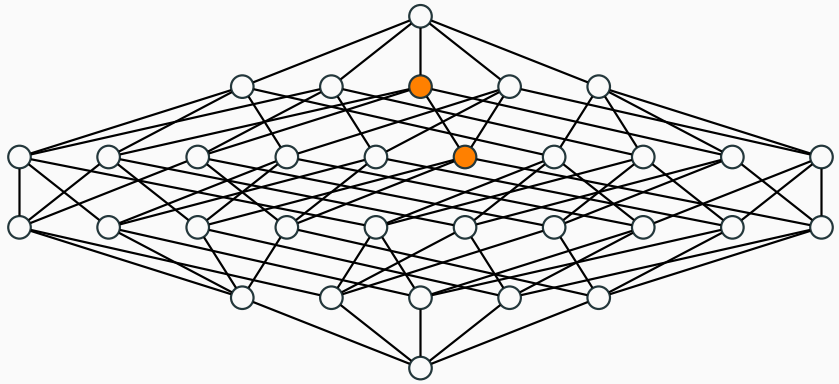
For any adjacent nodes v, v' of the graph, being given as input a database D, we can compute $\Pr(D \vDash q_v \vee q_{v'})$ in PTIME using only the negation, independence and disjunction rules.



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 $\Pr(D \models q_v \vee q_{v'})$ in PTIME using only the negation, independence and disjunction rules.

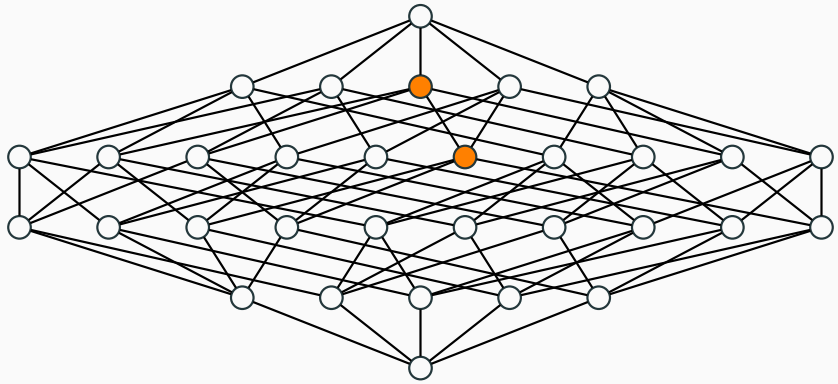


- **Idea**: starting from q, we will entirely uncolor the graph by using multiple times the following operations:
  - Uncolor two adjacent nodes that are colored
  - Color two adjacent nodes that were not colored

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 $\Pr(D \models q_v \vee q_{v'})$ in PTIME using only the negation, independence and disjunction rules.

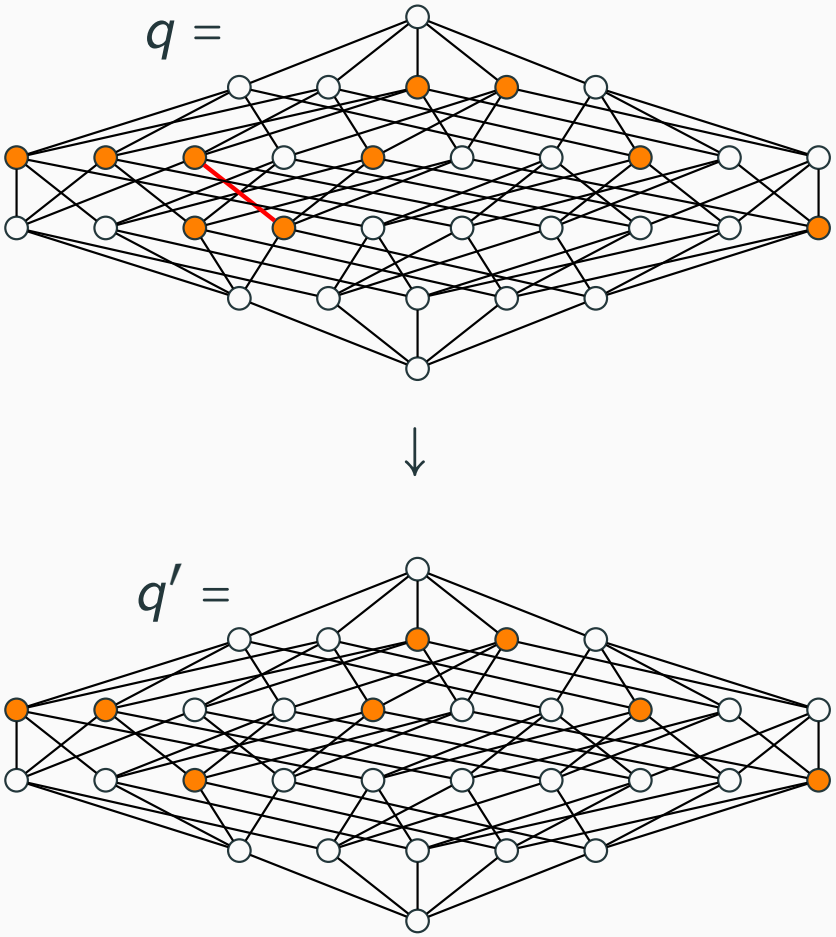


- **Idea**: starting from q, we will entirely uncolor the graph by using multiple times the following operations:
  - Uncolor two adjacent nodes that are colored
  - Color two adjacent nodes that were not colored

→ Simultaneously, we compute $\Pr(D \models q)$ using only the negation, independence and disjunction rules

Proof technique (3/4): computing $\Pr(D \models q)$

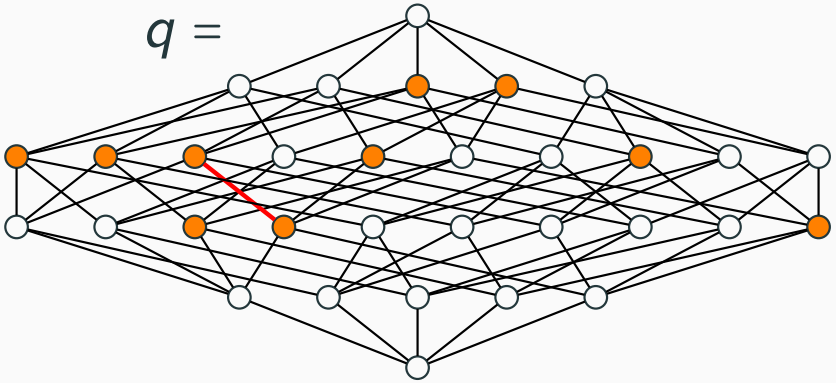
Uncoloring:



Proof technique (3/4): computing $\Pr(D \models q)$

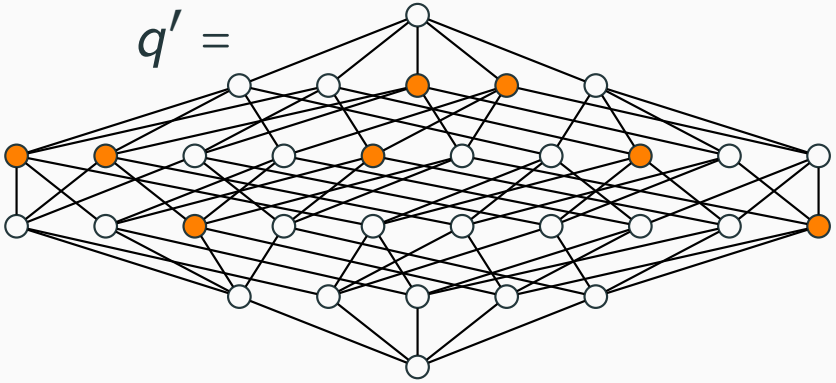
Uncoloring:

$q =$



$\downarrow$

$q' =$

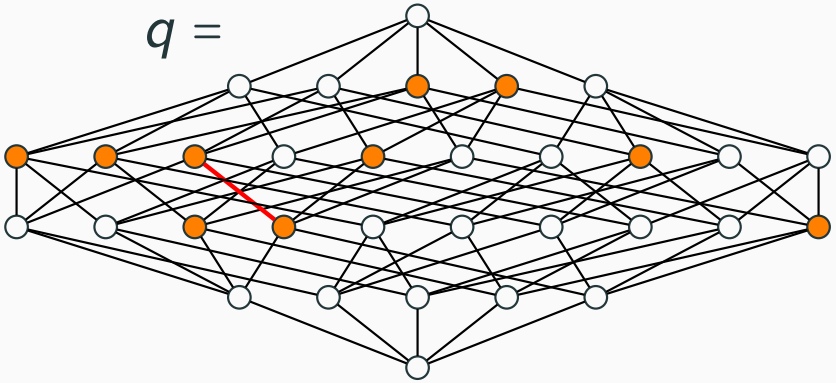


$$\Pr(D \models q) = \Pr(D \models q_v \vee q_{v'}) + \Pr(D \models q')$$

Proof technique (3/4): computing $\Pr(D \models q)$

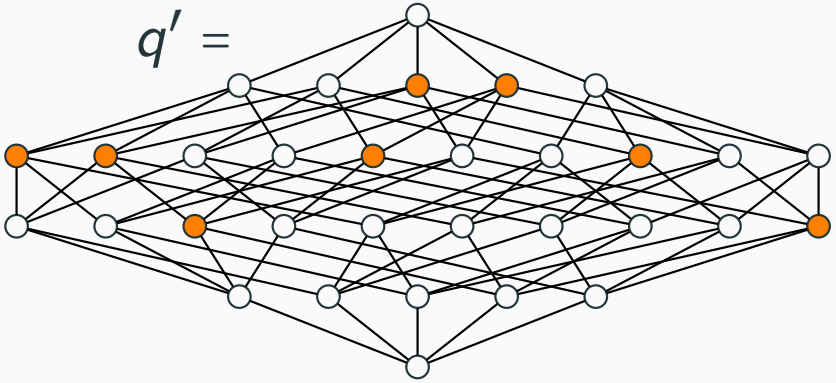
Uncoloring:

$q =$



$\downarrow$

$q' =$



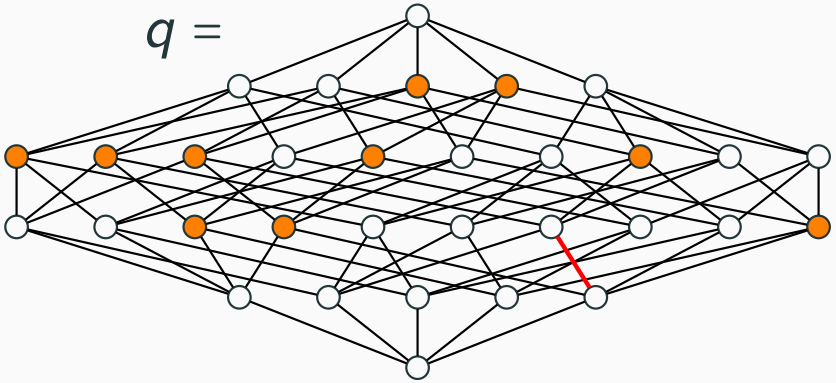
$$\Pr(D \models q) = \Pr(D \models q_v \vee q_{v'}) + \Pr(D \models q')$$

Then continue with q'

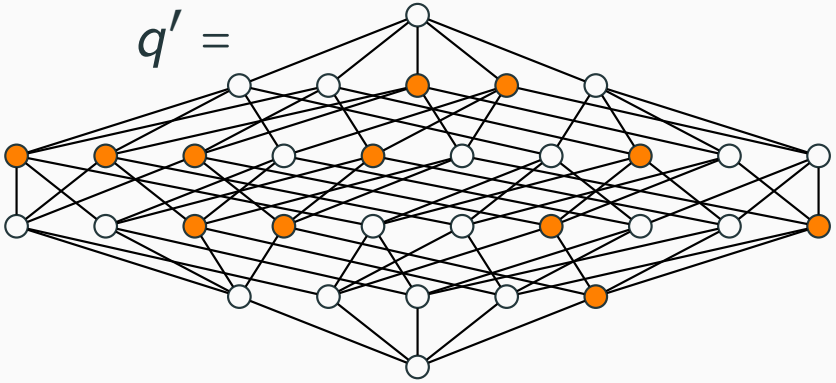
Proof technique (3/4): computing $\Pr(D \models q)$

Coloring:

$q =$



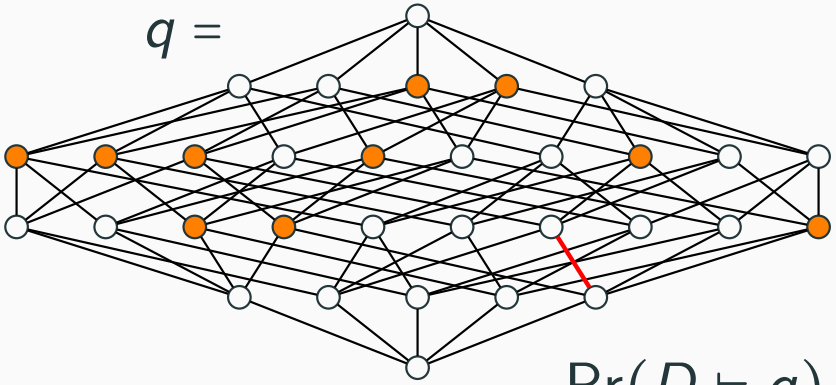
$q' =$



Proof technique (3/4): computing $\Pr(D \models q)$

Coloring:

$q =$

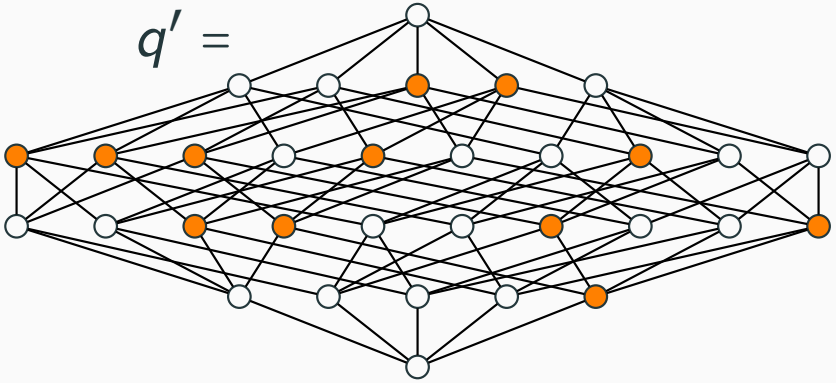


$$\Pr(D \models q) = 1 - \Pr(D \models \neg q)$$

$\downarrow$

$$= 1 - [\Pr(D \models q_v \vee q_{v'}) + \Pr(D \models \neg q')]$$

$q' =$

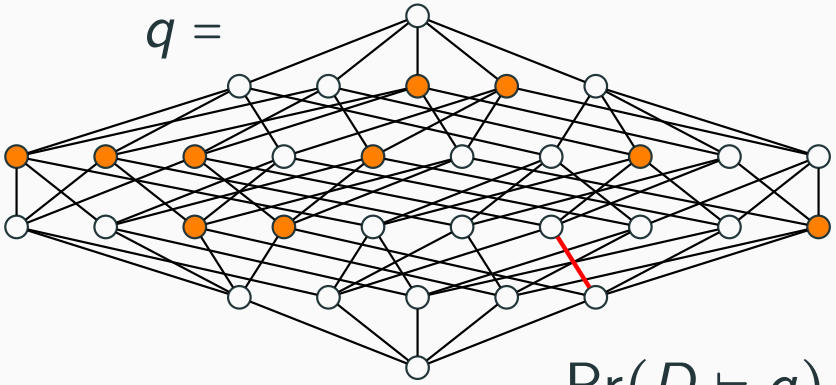


$$= 1 - [\Pr(D \models q_v \vee q_{v'}) + (1 - \Pr(D \models q'))]$$

Proof technique (3/4): computing $\Pr(D \models q)$

Coloring:

$q =$

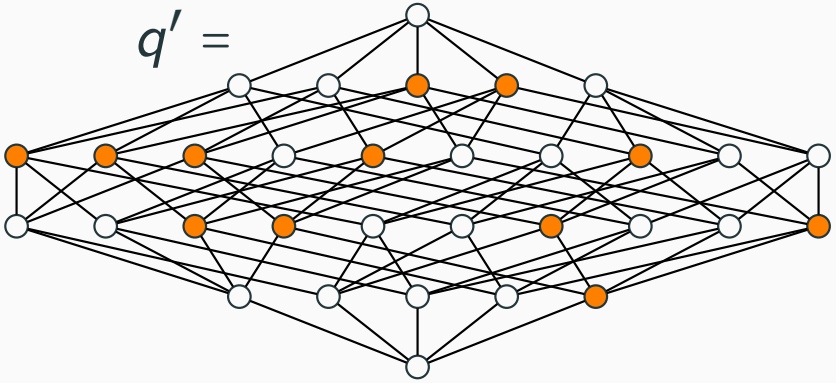


$$\Pr(D \models q) = 1 - \Pr(D \models \neg q)$$

$$= 1 - [\Pr(D \models q_v \vee q_{v'}) + \Pr(D \models \neg q')]$$

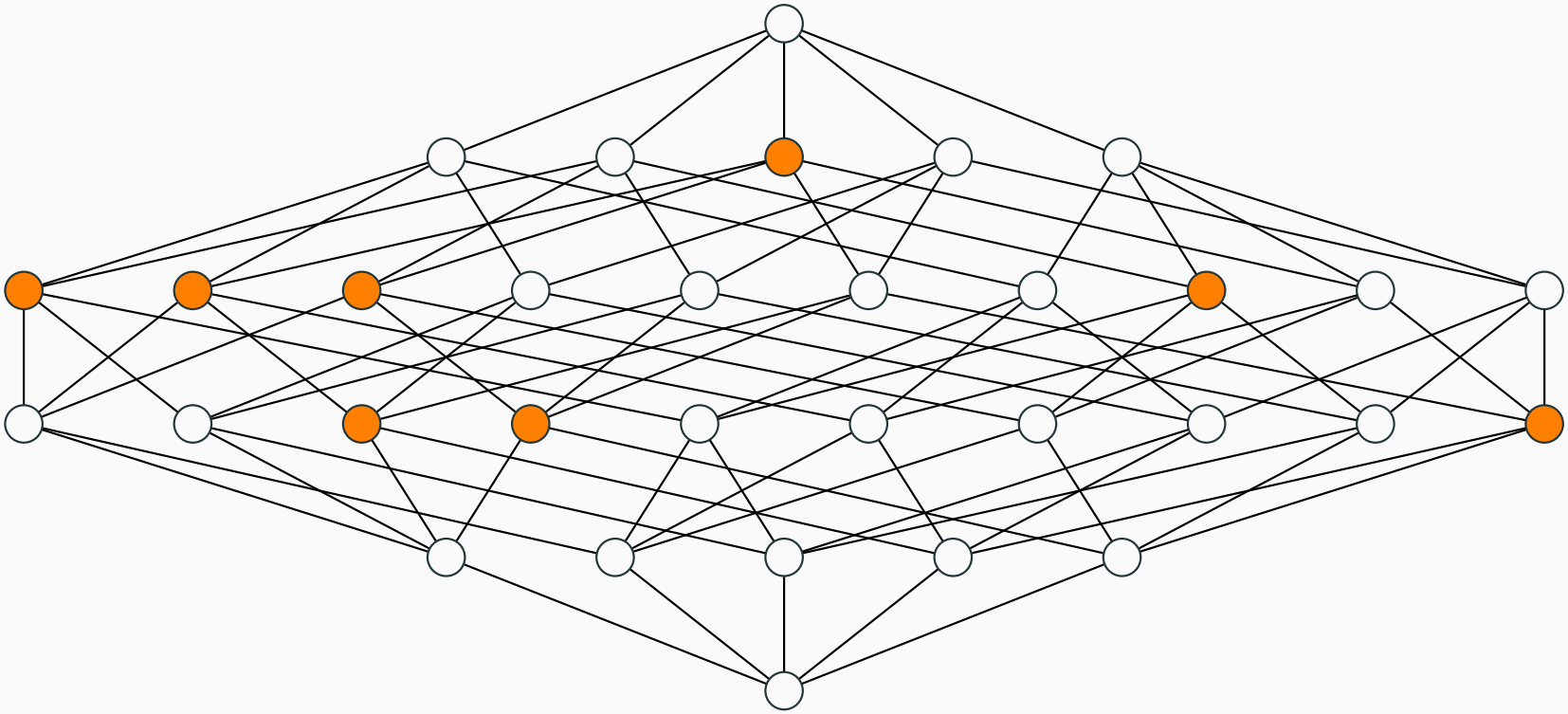
$$= 1 - [\Pr(D \models q_v \vee q_{v'}) + (1 - \Pr(D \models q'))]$$

$q' =$

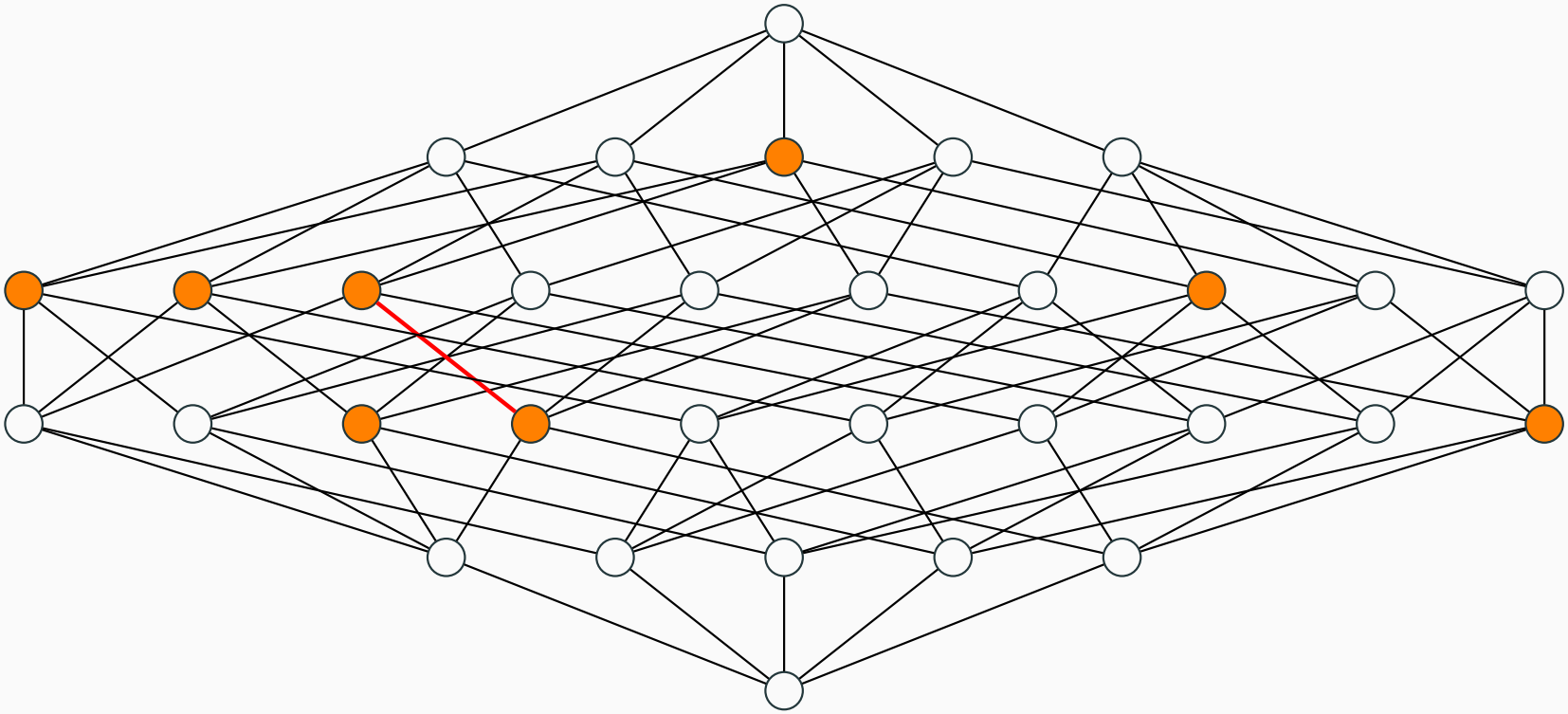


Then continue with q'

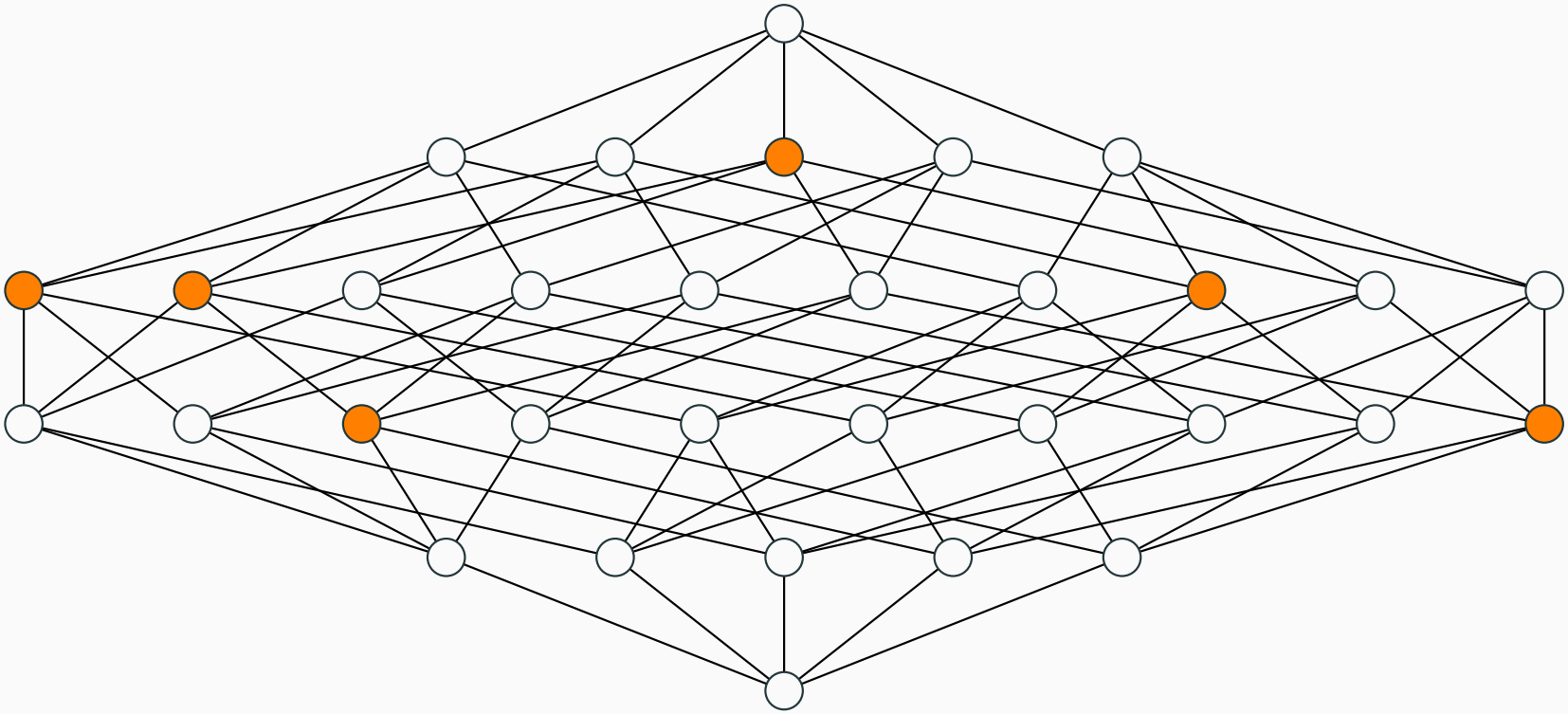
Proof technique (4/4): how to uncolor the graph?



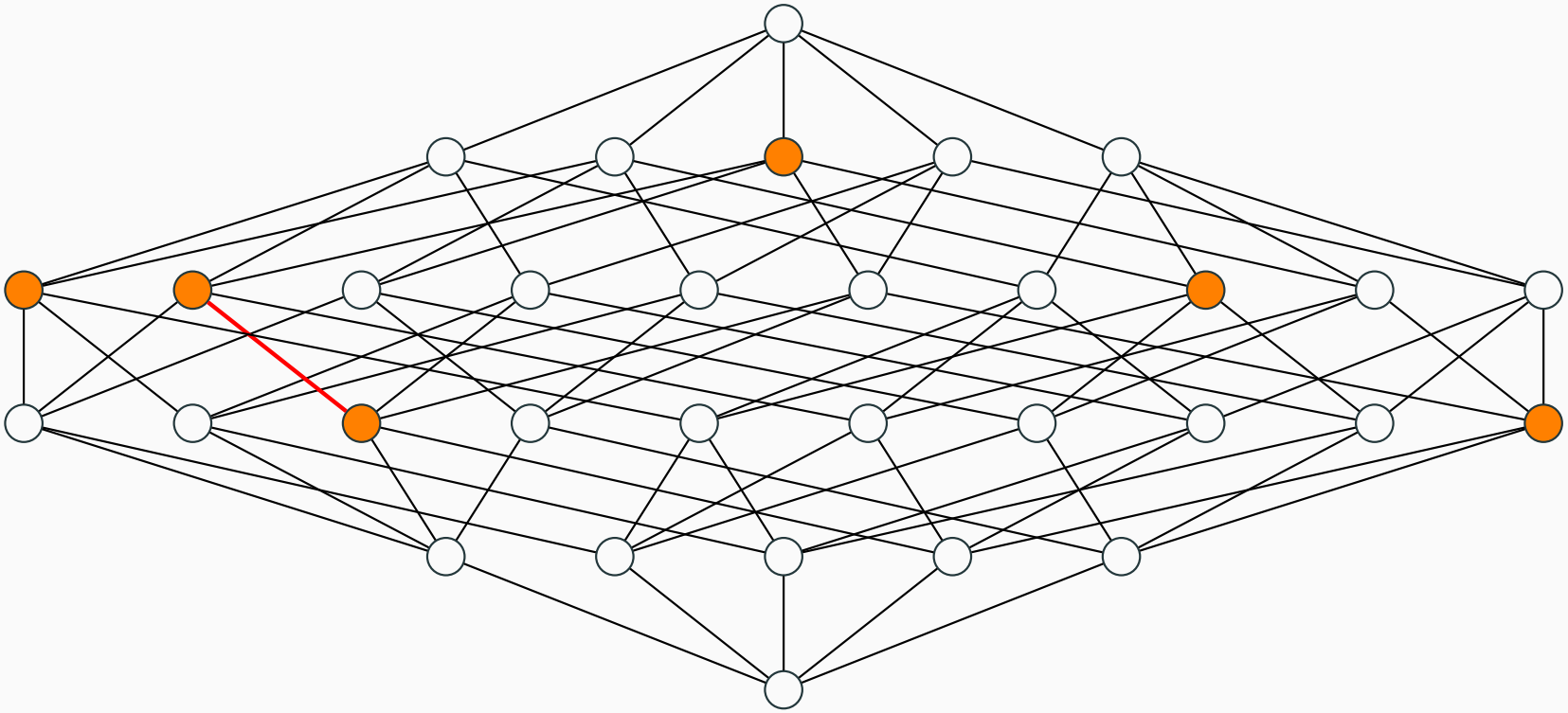
Proof technique (4/4): how to uncolor the graph?



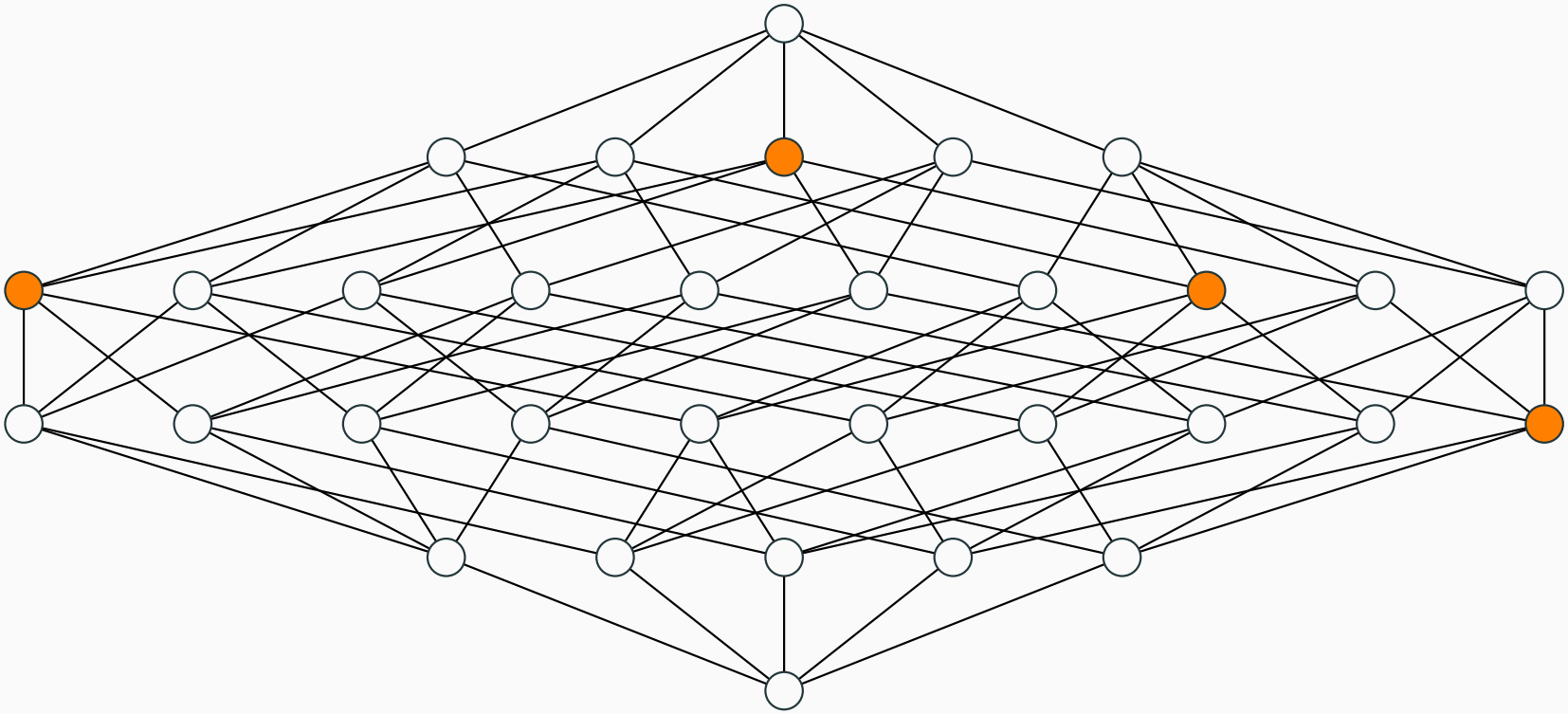
Proof technique (4/4): how to uncolor the graph?



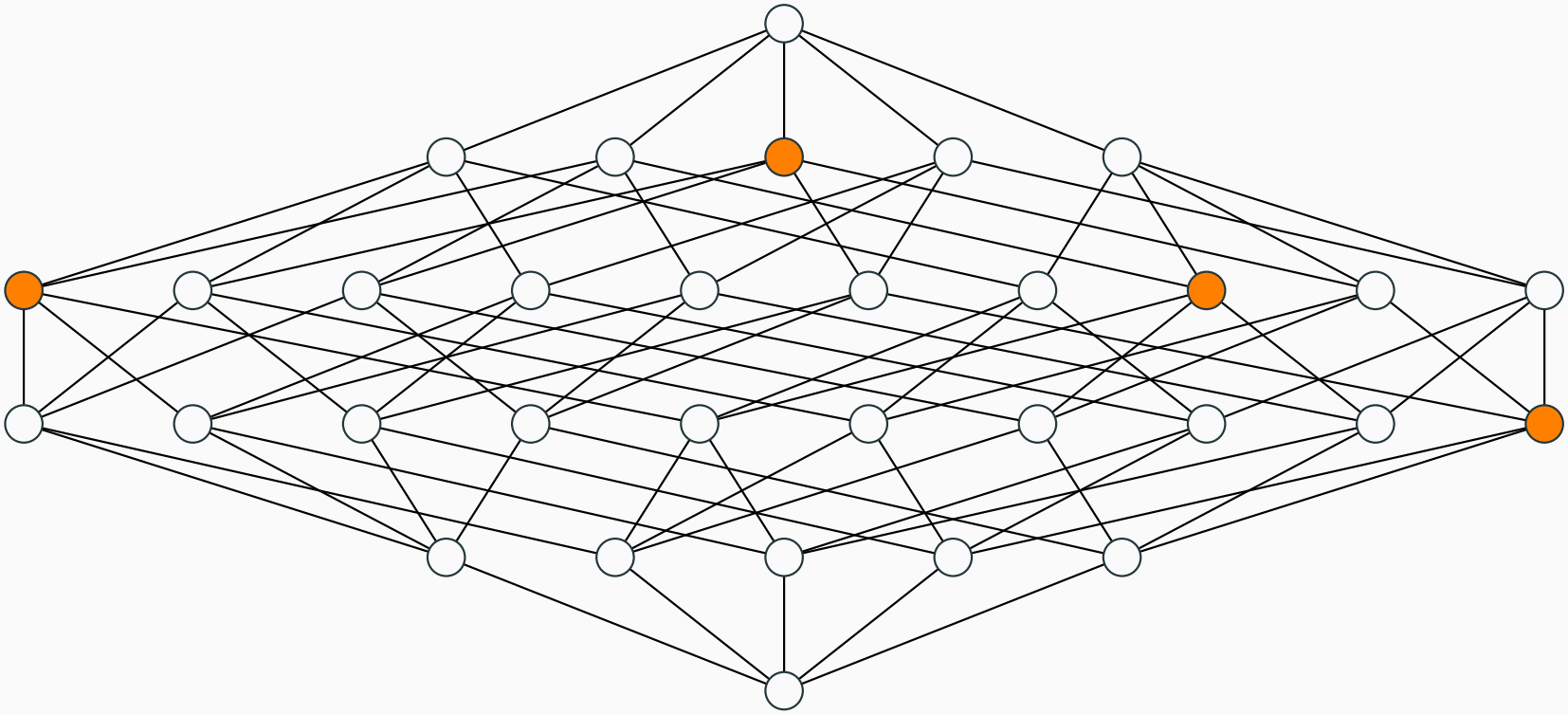
Proof technique (4/4): how to uncolor the graph?



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



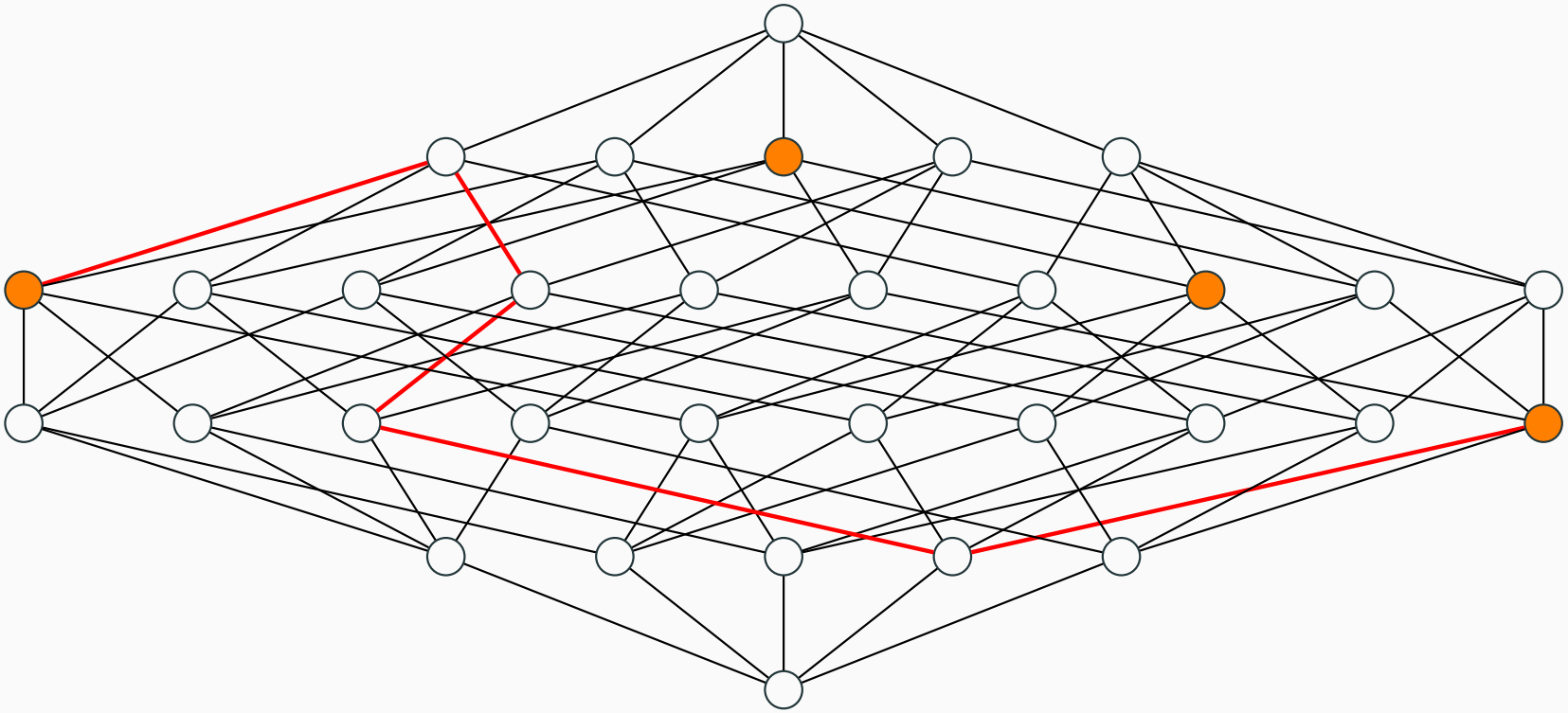
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

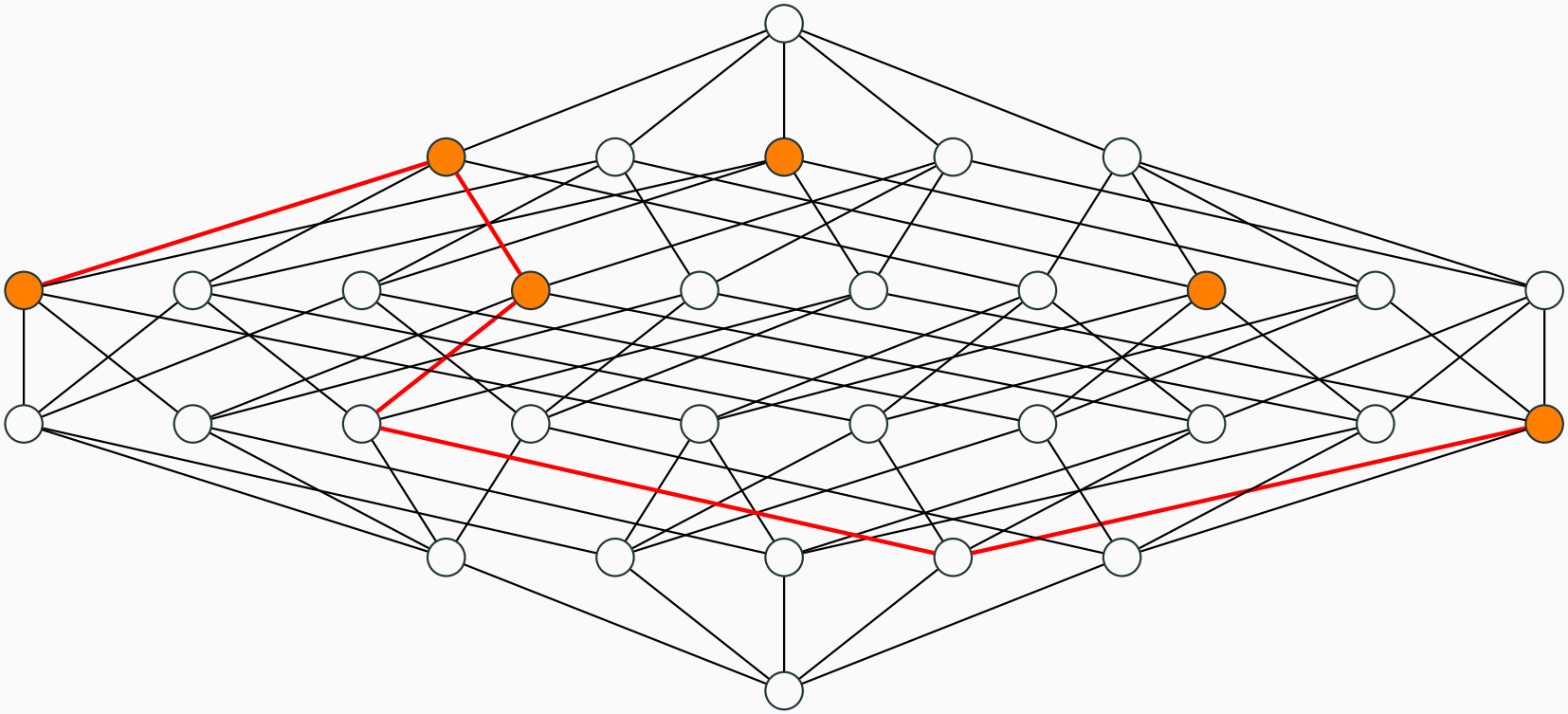
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

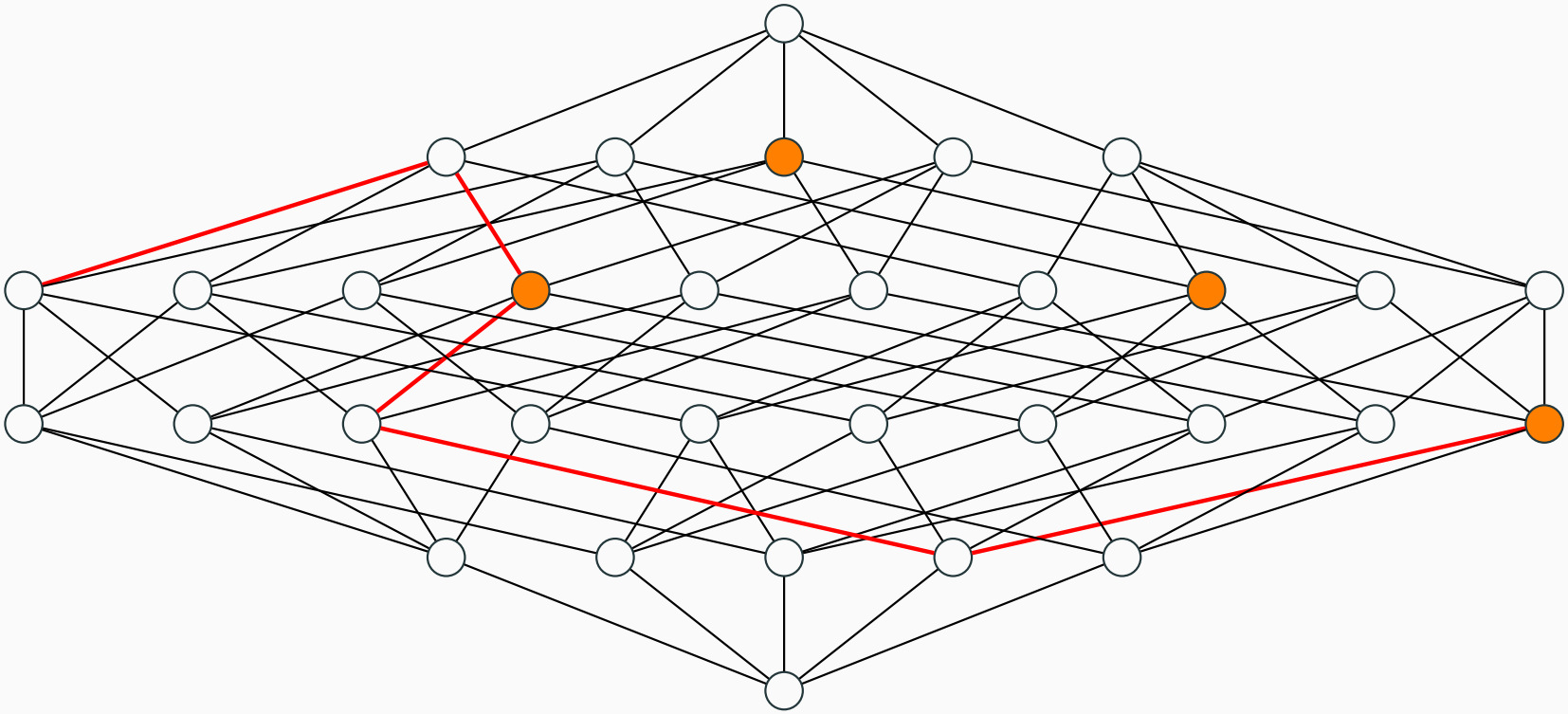
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

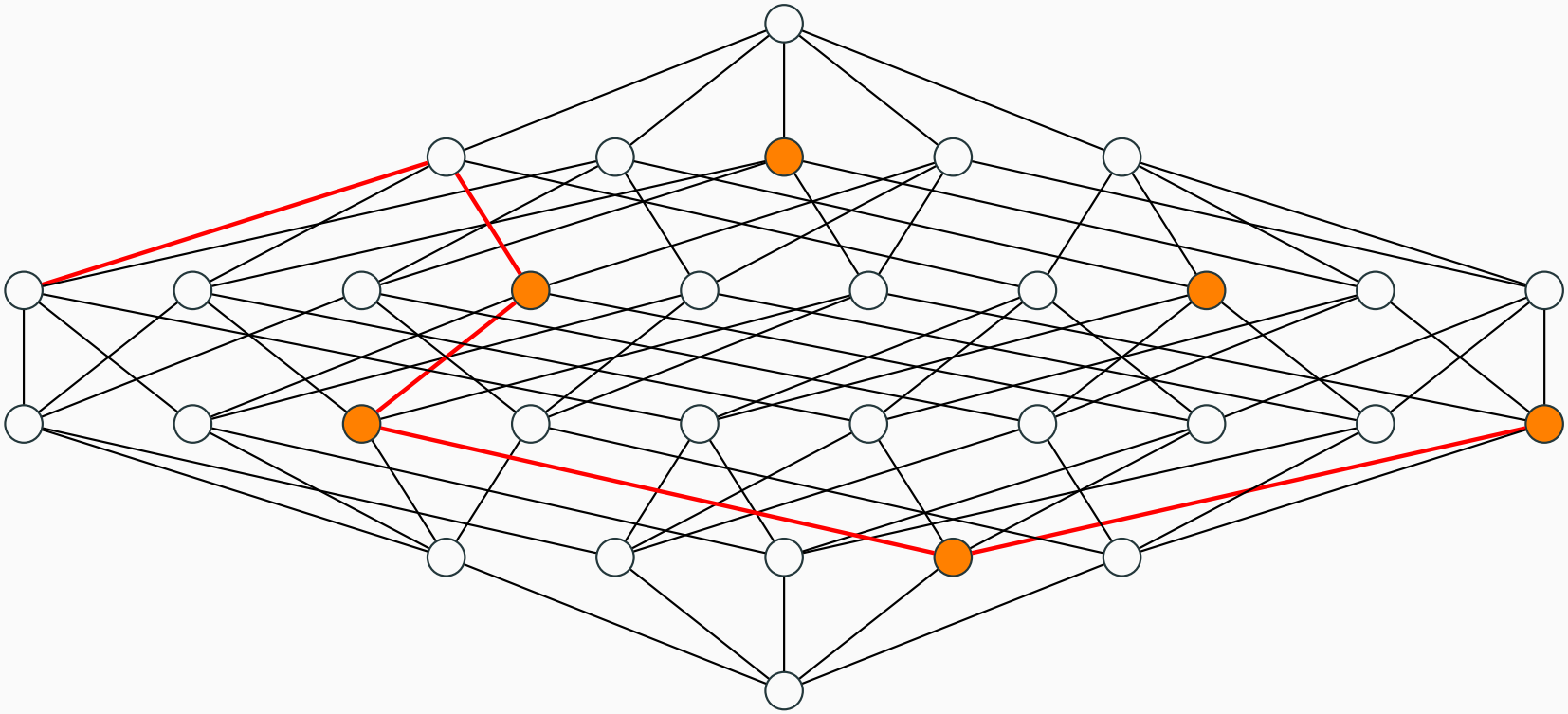
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

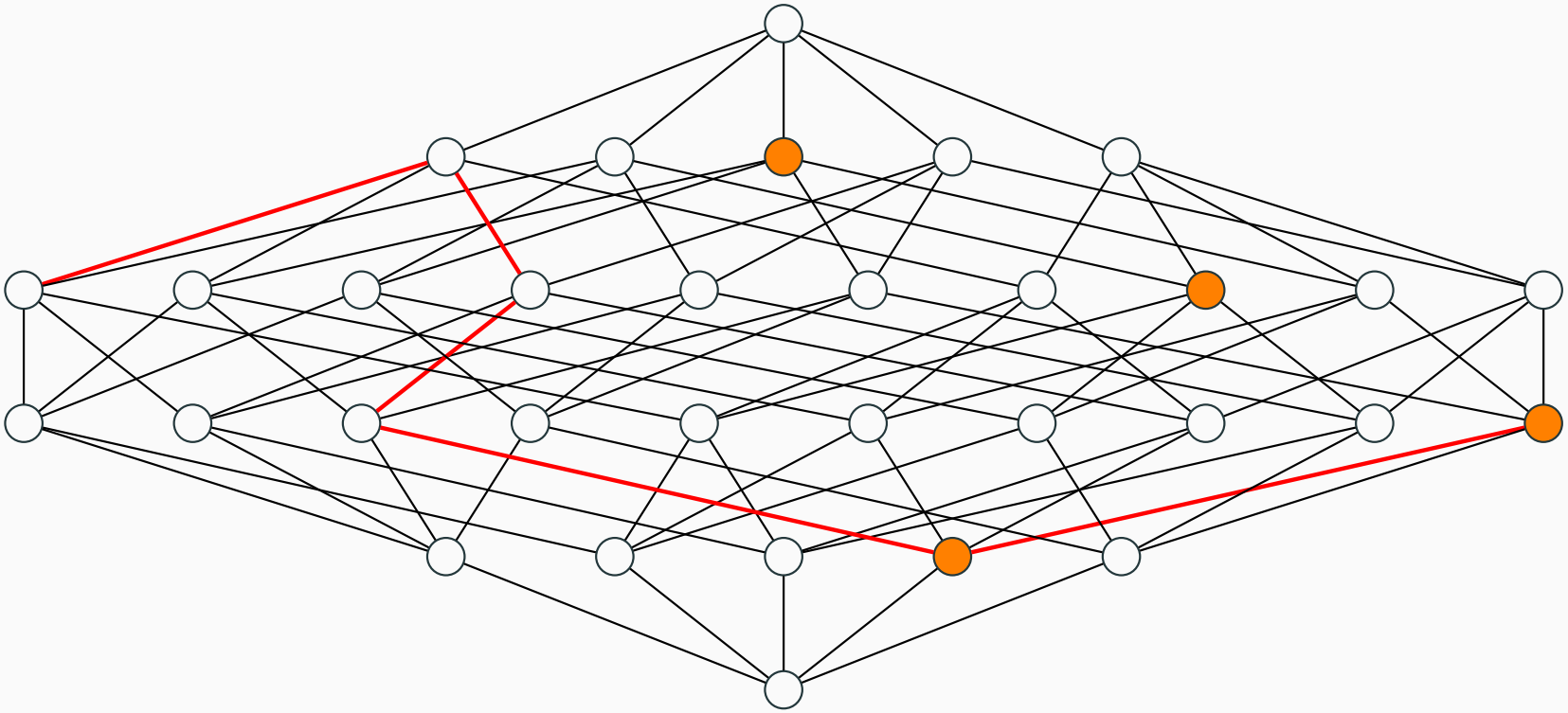
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

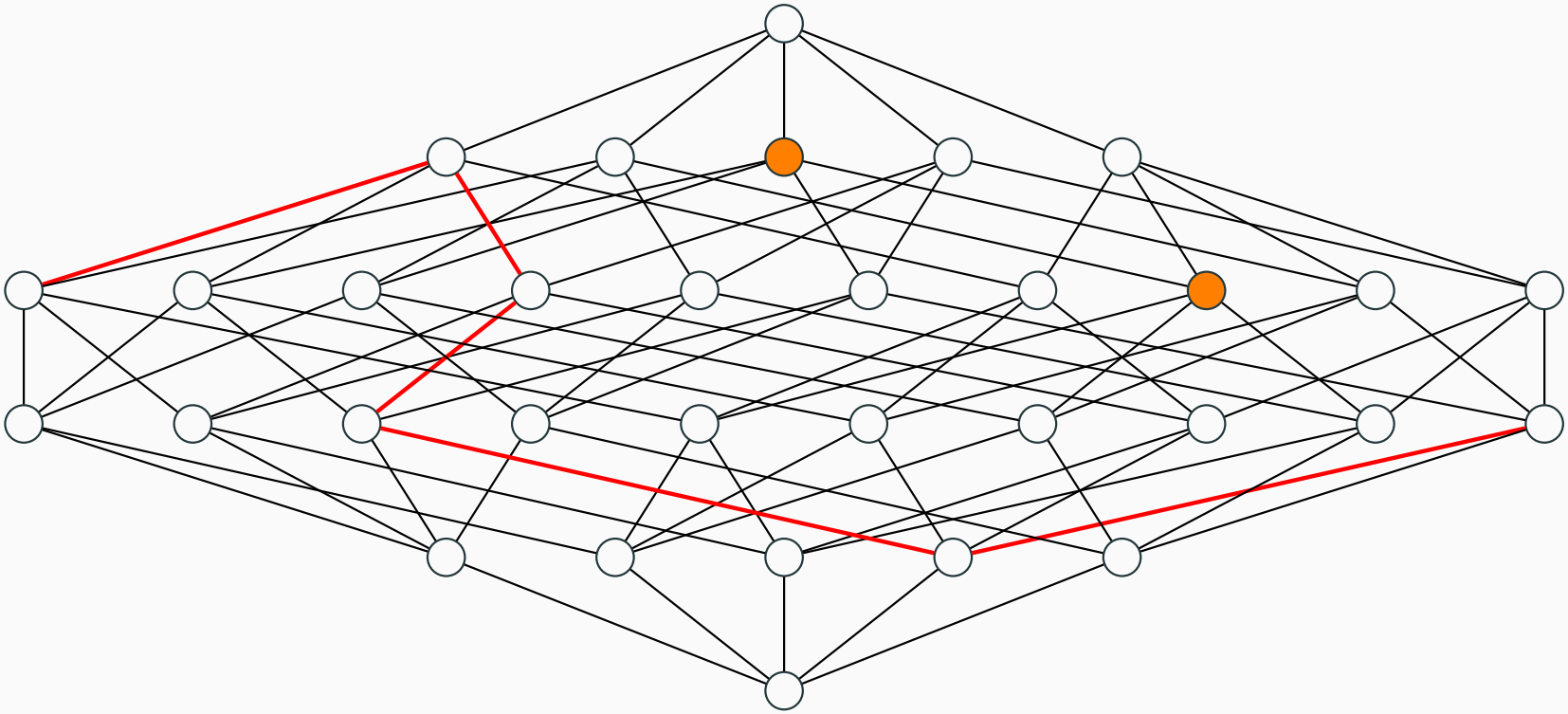
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

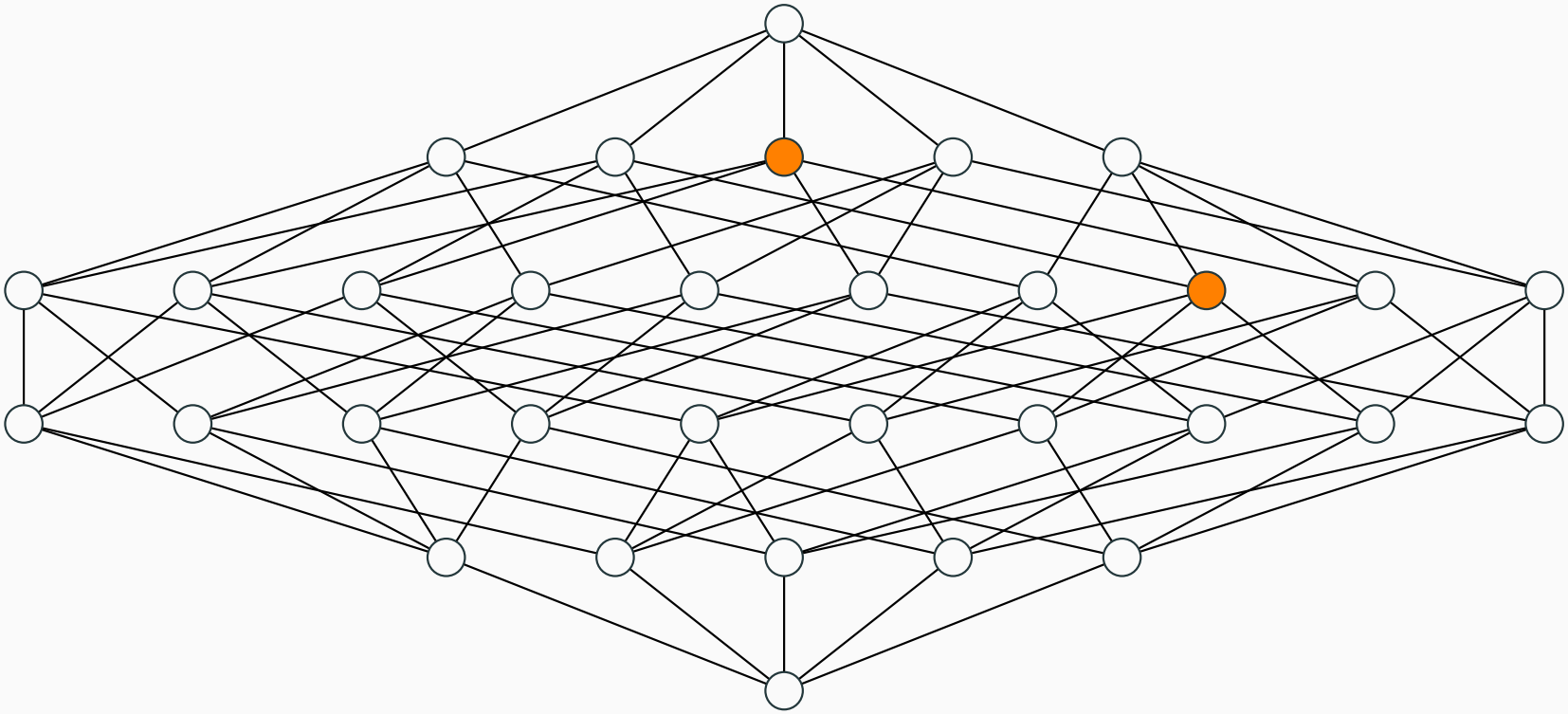
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

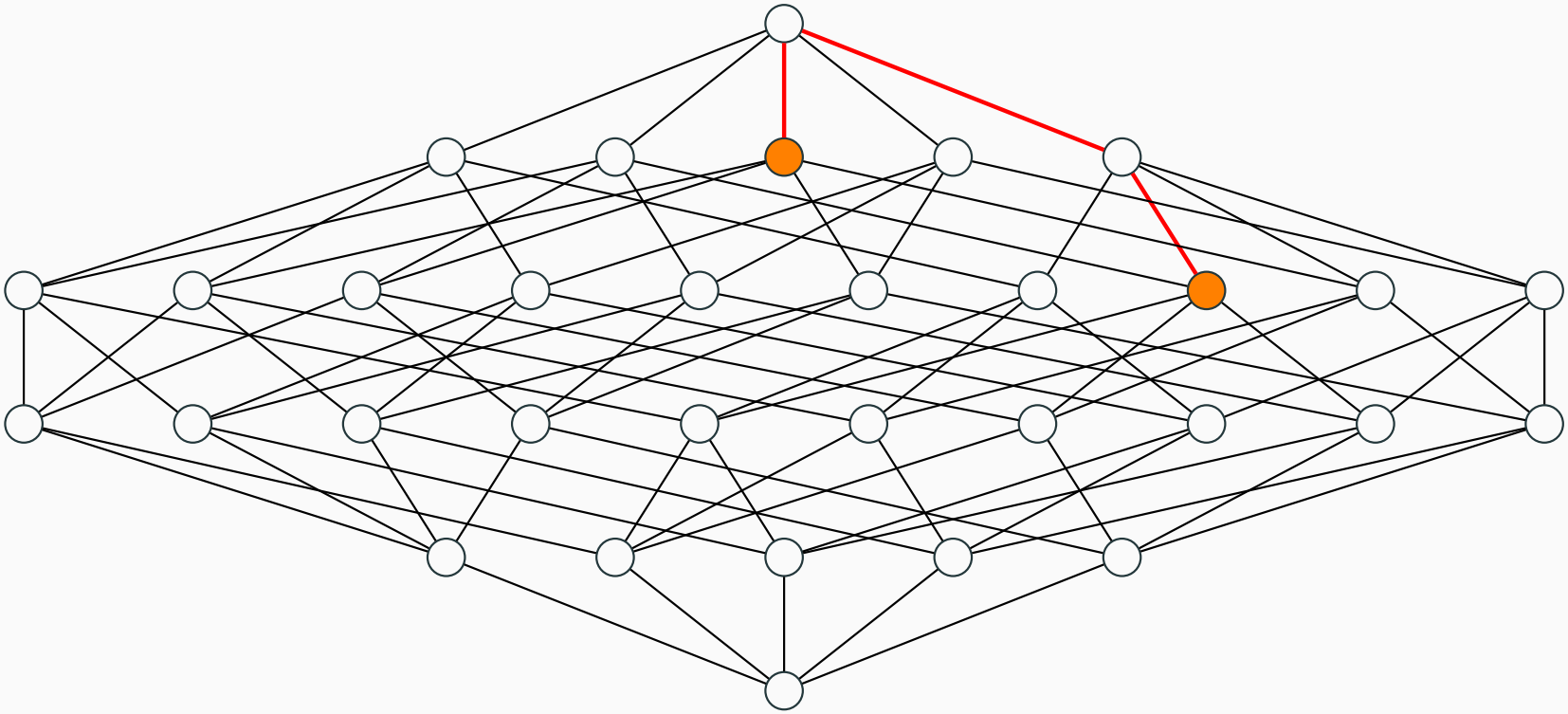
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

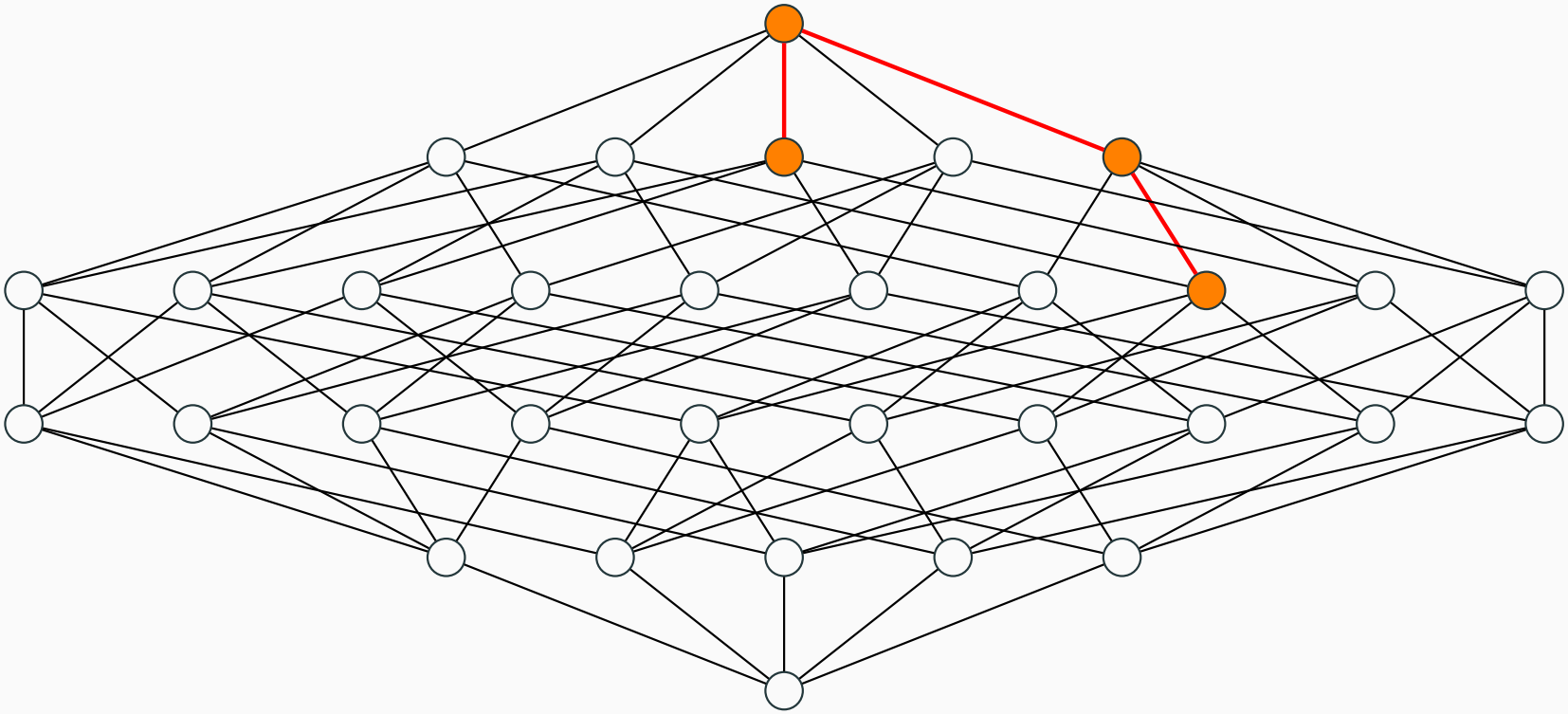
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

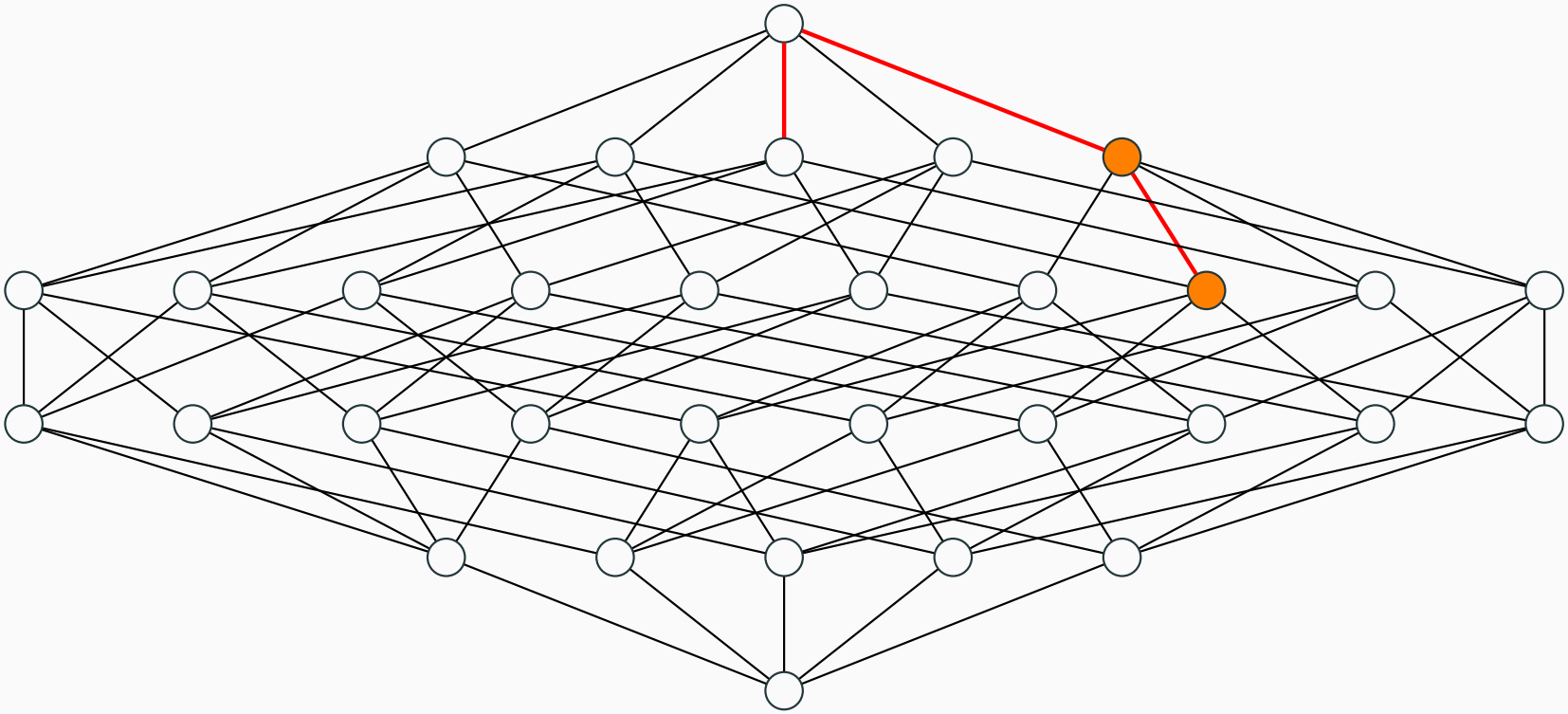
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

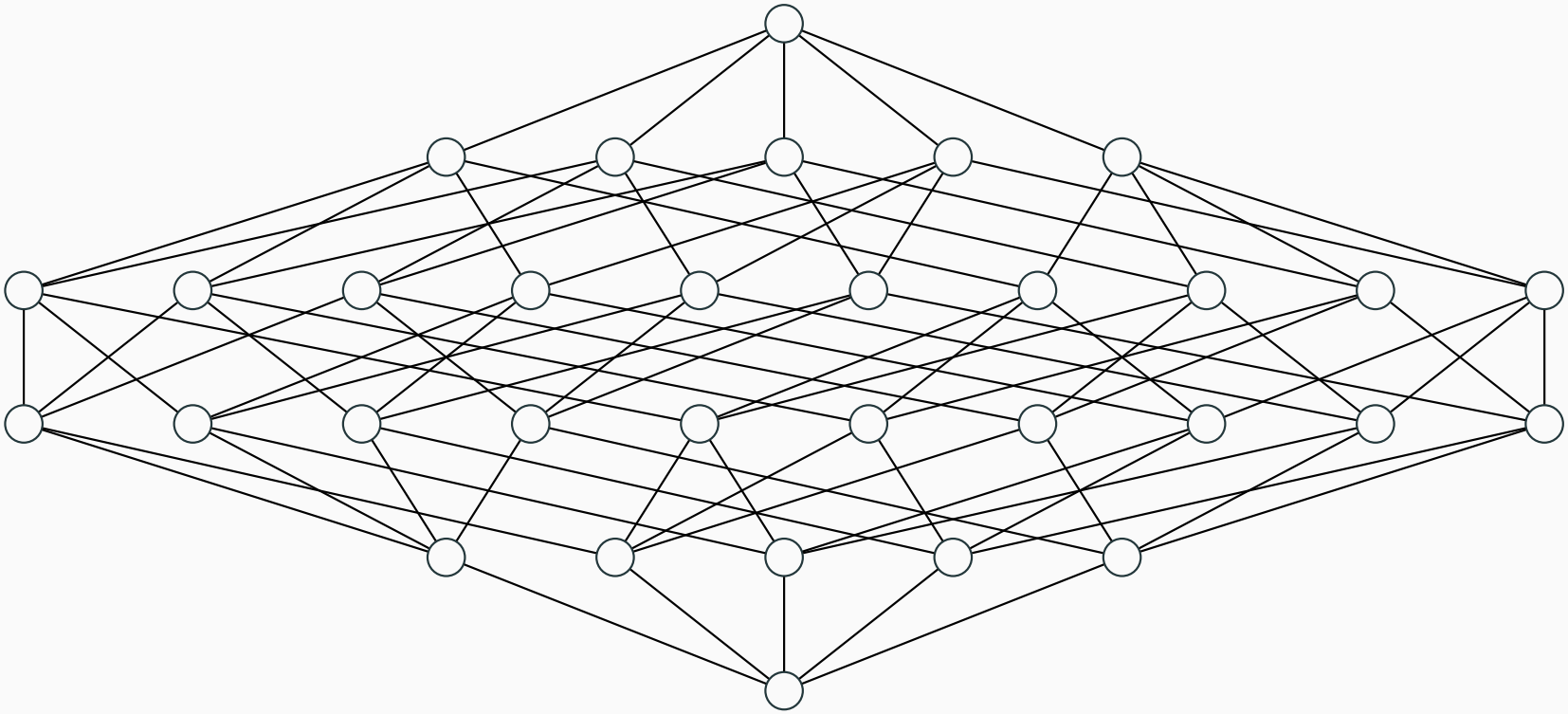
Proof technique (4/4): how to uncolor the graph?



Proposition (us)

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

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



Proposition (us)

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

Conclusion

- The $\mathcal{H}$ queries were the first obstacle to showing that PQE for safe queries can be handled with knowledge compilation techniques, because Dalvi and Suciu's algorithm for these queries must use the **inclusion–exclusion rule**
- We showed that for these queries, we can replace the inclusion–exclusion rule by the **disjunction rule**, thus capturing the tractability of $\mathrm{PQE}(q)$ with circuit classes from knowledge compilation
- Some credit goes to Dan Olteanu and Guy Van den Broeck
- Currently working (with Antoine Amarilli and Louis Jachiet) on extending this new technique to handle all UCQs.
  - → we have some nice combinatorial open problems: see here and there!

Thanks for your attention!

Bibliography I

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.