# A Game on Directed Acyclic Graphs

Antoine Amarilli[1], Louis Jachiet[1] and **Mikaël Monet**[2]

November 14th, 2019

[1]Télécom Paris, France
[2]Millennium Institute for Foundational Research on Data, Chile

# Motivation

- Dalvi and Suciu's algorithm for computing the probability of a safe UCQ uses essentially three rules [2]:
  - Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when $A, B$ independent
  - Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - Inclusion-exclusion: $\Pr(A \vee B \vee C) = \Pr(A) + \Pr(B) + \Pr(C) - \Pr(A \wedge B) - \Pr(A \wedge C) - \Pr(B \wedge C) + \Pr(A \wedge B \wedge C)$

# Motivation

- Dalvi and Suciu's algorithm for computing the probability of a safe UCQ uses essentially three rules [2]:
  - Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when $A, B$ independent
  - Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - Inclusion-exclusion: $\Pr(A \vee B \vee C) = \Pr(A) + \Pr(B) + \Pr(C) - \Pr(A \wedge B) - \Pr(A \wedge C) - \Pr(B \wedge C) + \Pr(A \wedge B \wedge C)$
- Knowledge Compilation-based techniques use three rules:
  - Independence, negation
  - Disjoint disjunction: $\Pr(A \vee B) = \Pr(A) + \Pr(B)$ when $A, B$ disjoint

# Motivation

- Dalvi and Suciu's algorithm for computing the probability of a safe UCQ uses essentially three rules [2]:
  - Independence: $\Pr(A \wedge B) = \Pr(A) \times \Pr(B)$ when $A, B$ independent
  - Negation: $\Pr(\neg A) = 1 - \Pr(A)$
  - Inclusion-exclusion: $\Pr(A \vee B \vee C) = \Pr(A) + \Pr(B) + \Pr(C) - \Pr(A \wedge B) - \Pr(A \wedge C) - \Pr(B \wedge C) + \Pr(A \wedge B \wedge C)$
- Knowledge Compilation-based techniques use three rules:
  - Independence, negation
  - Disjoint disjunction: $\Pr(A \vee B) = \Pr(A) + \Pr(B)$ when $A, B$ disjoint

An open problem in probabilistic databases: can we handle all safe UCQs using Knowledge Compilation? (via so-called deterministic decomposable circuits (d-Ds))

# The game

(Invented with Antoine Amarilli)

- We consider directed acyclic graphs (DAGs) where nodes can be "**on**" or "**off**"

→ Let $G = (V, E)$ be a DAG with all nodes being initially off

- **Goal:** switch on all the nodes using a sequence of moves, where a move can be:

# The game

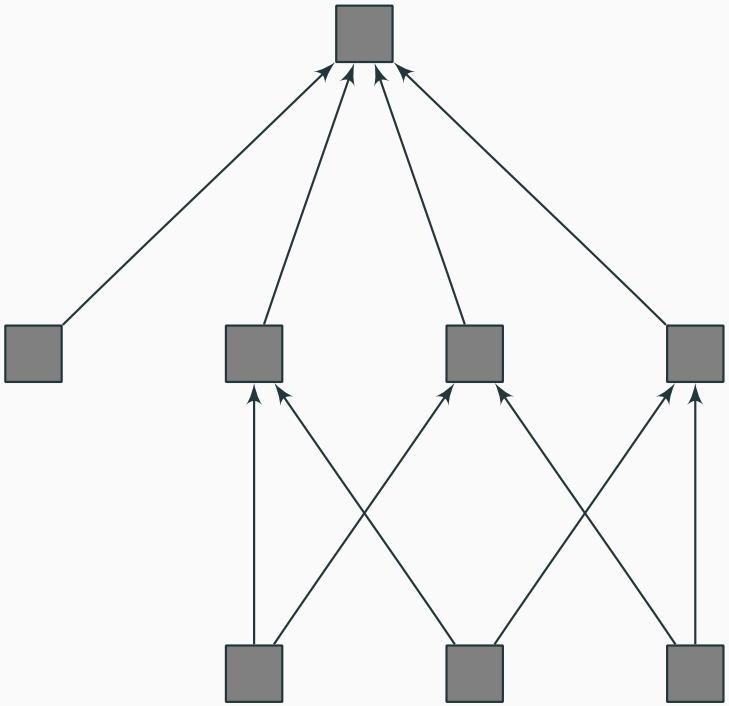
(Invented with Antoine Amarilli)

- We consider directed acyclic graphs (DAGs) where nodes can be "on" or "off"

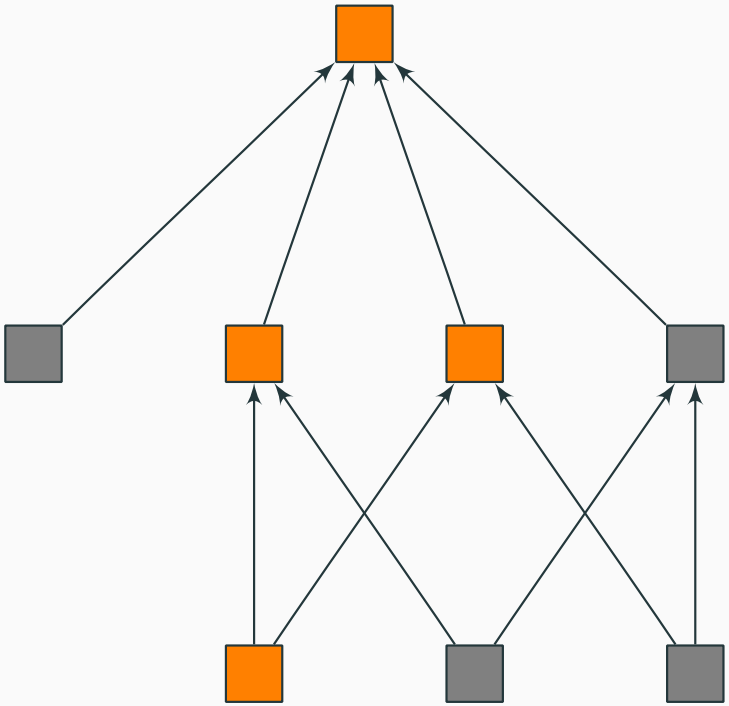
→ Let $G = (V, E)$ be a DAG with all nodes being initially off

- **Goal:** switch on all the nodes using a sequence of moves, where a move can be:
  - pick a node $u \in V$. If $u$ and all its descendants are off, then we can switch them all on; we say that we have played an on-move on $u$
  - pick a node $u \in V$. If $u$ and all its descendants are on, then we can switch them all off; we say that we have played an off-move on $u$

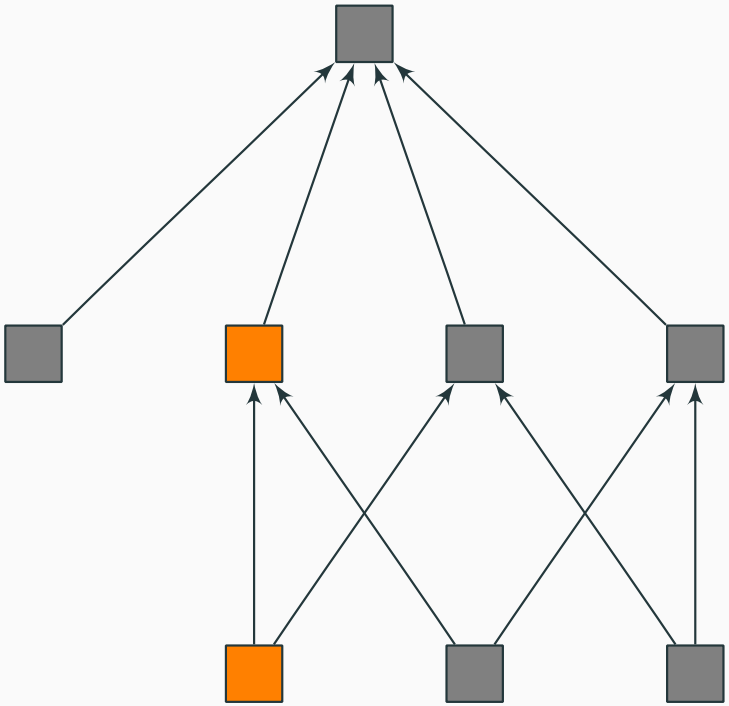
# Example



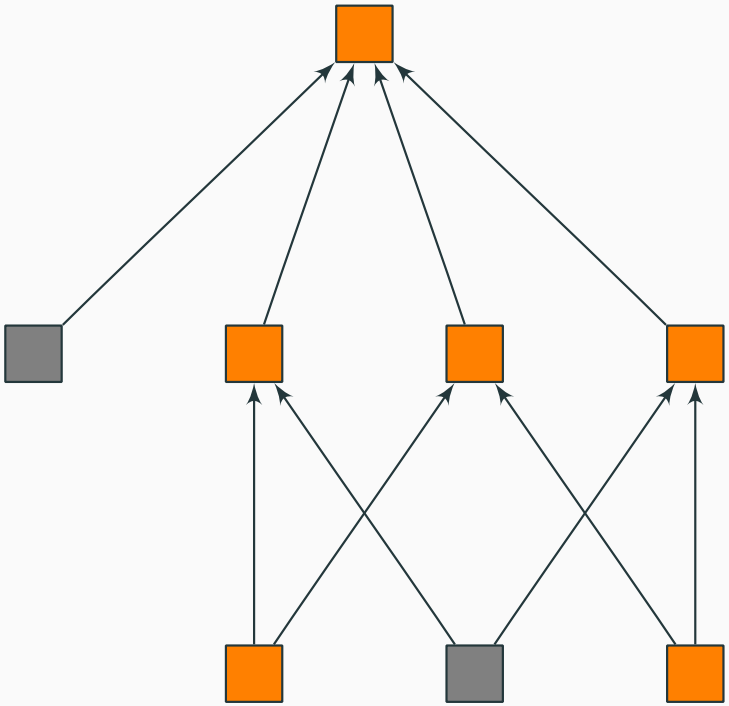
# Example



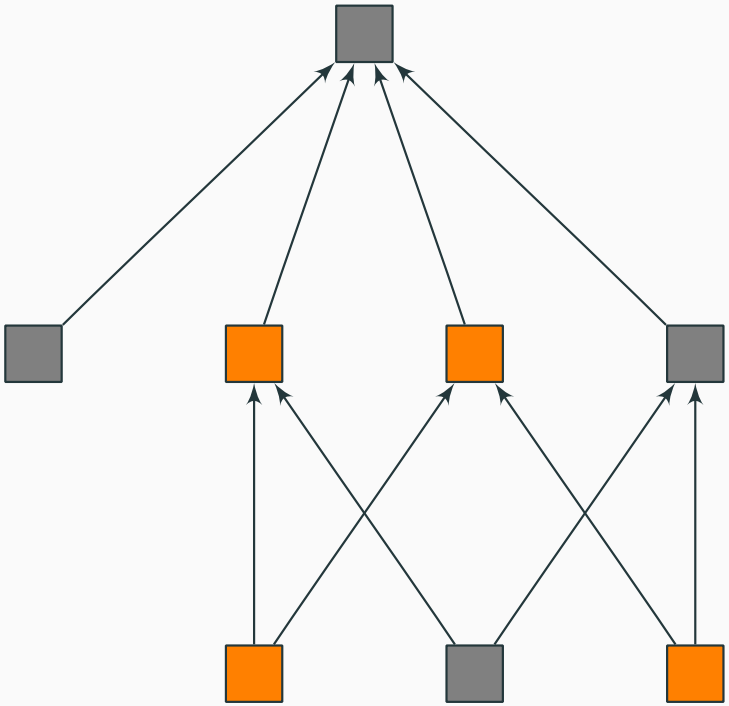
# Example



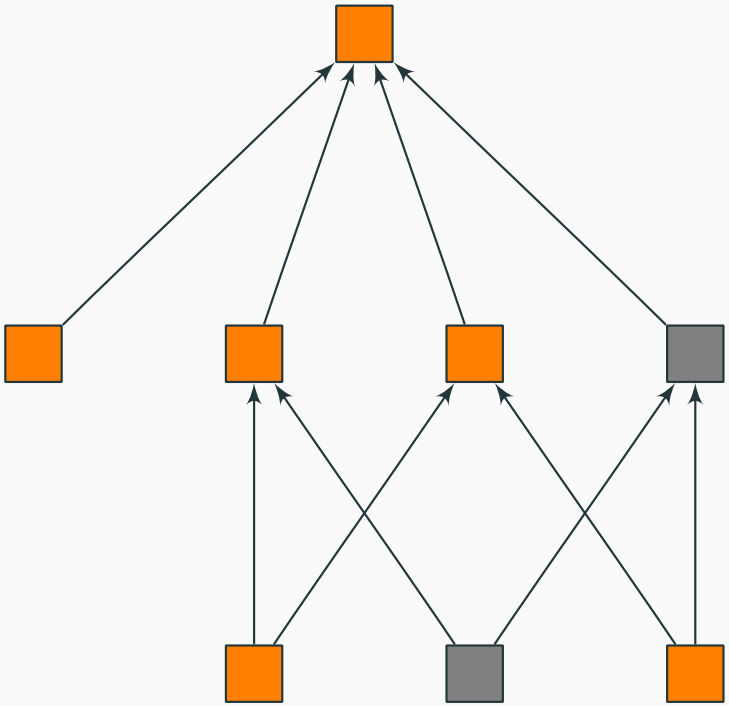
# Example



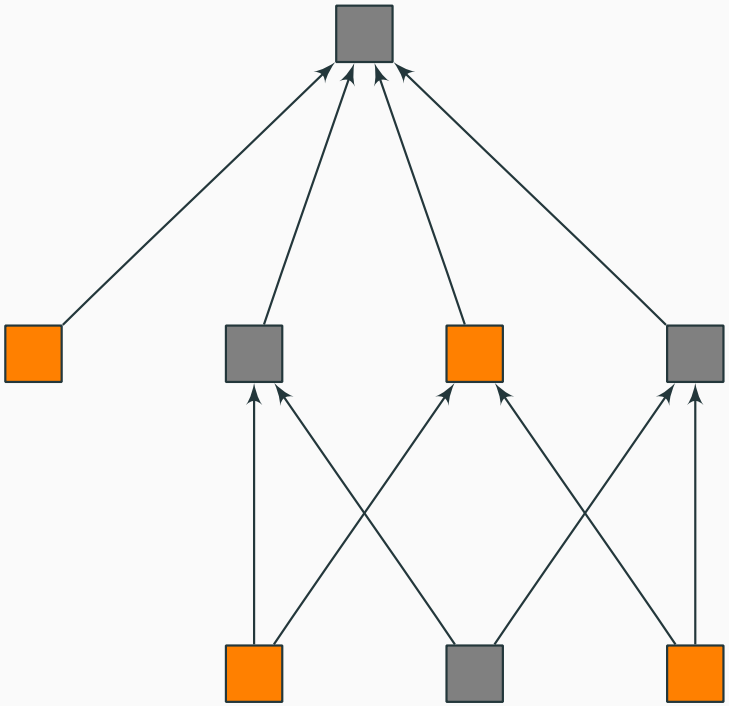
# Example



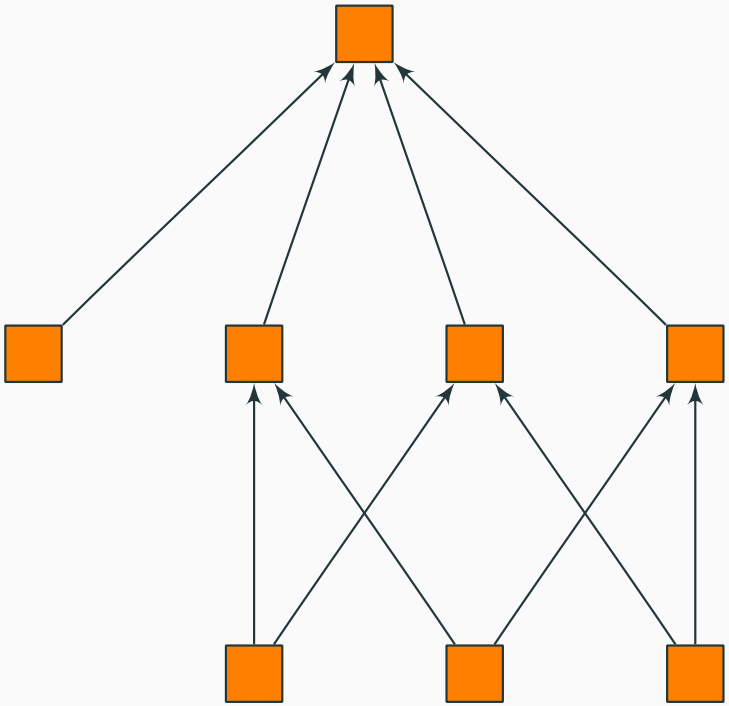
# Example



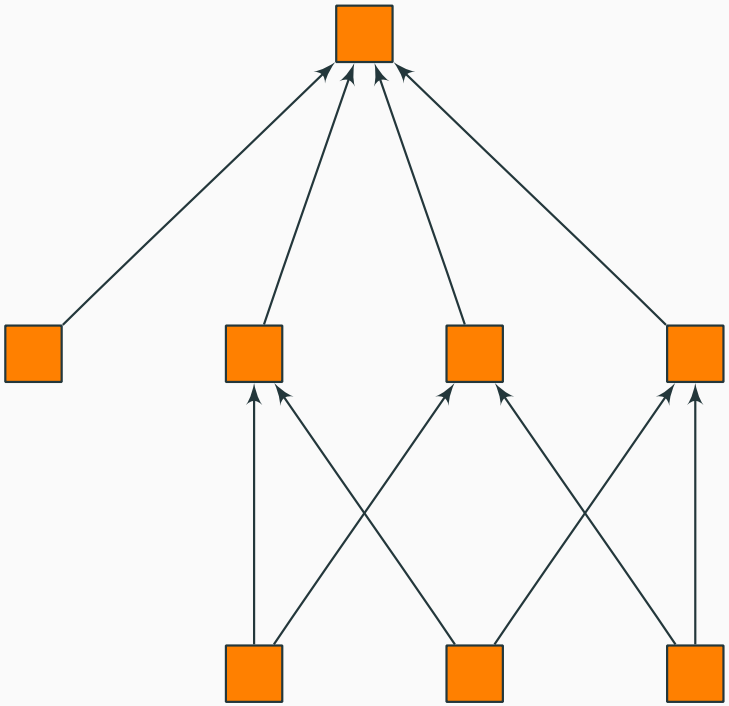
# Example



# Example

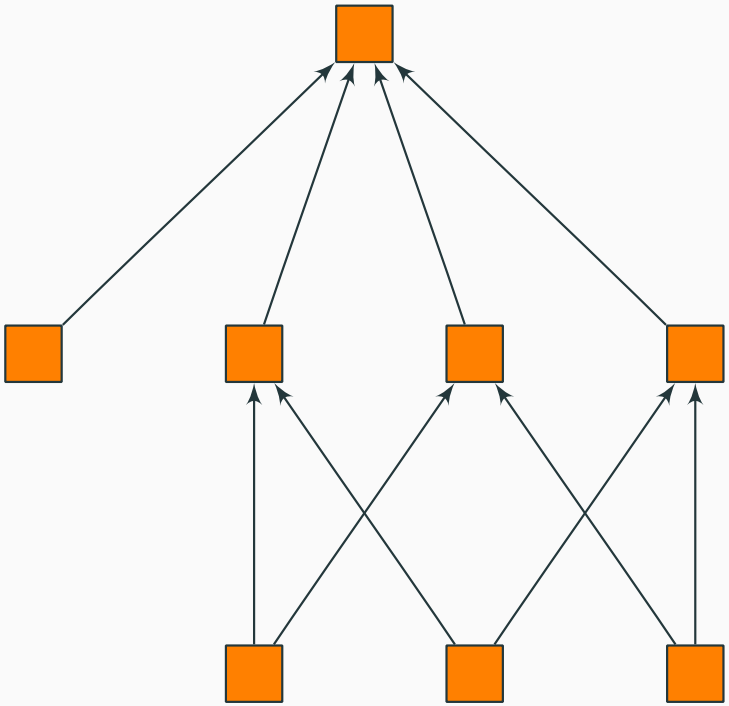


# Example



WIN!

# Example



WIN!

Can be used to re-order terms in the inclusion-exclusion formula in a disjoint manner

# Question

- Can we win for every DAGs?

# Question

- Can we win for every DAGs?

$\rightarrow$ Yes (easy top-down induction)

# Question

- Can we win for every DAGs?

$\rightarrow$ Yes (easy top-down induction)

- Now add the following restriction: for every node $u$ then:
  - we only play on-moves on $u$; or
  - we only play off-moves on $u$

# Question

- Can we win for every DAGs?

$\rightarrow$ Yes (easy top-down induction)

- Now add the following restriction: for every node $u$ then:
  - we only play on-moves on $u$; or
  - we only play off-moves on $u$
- Can we win for all the DAGs?

# Question

- Can we win for every DAGs?

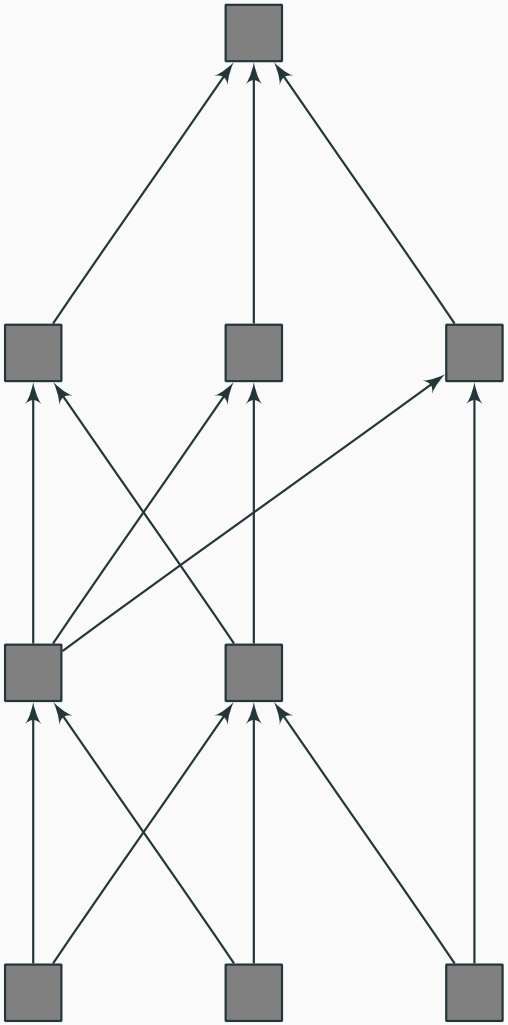
$\rightarrow$ Yes (easy top-down induction)

- Now add the following restriction: for every node $u$ then:
  - we only play on-moves on $u$; or
  - we only play off-moves on $u$
- Can we win for all the DAGs?

$\rightarrow$ No

# Counter example

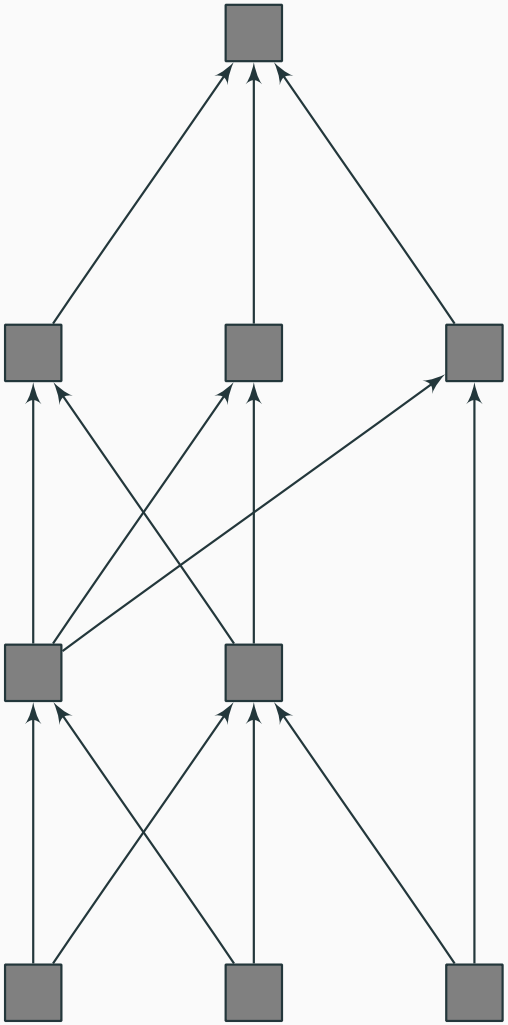
Computer search by Louis Jachiet gave us this DAG



We cannot win on this DAG

# Counter example

Computer search by Louis Jachiet gave us this DAG



We cannot win on this DAG

→ Characterize the winning DAGs

→ For our purposes (constructing d-Ds for safe UCQs): Do we win when $G$ is a join semi-lattice?

- See also [1, 3]

Thanks for your attention!

# Bibliography I



a3nm.
**Lighting up all elements of a poset by toggling upsets, 2019.**


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


Mikaël Monet.
**Perfect matching of monotone Boolean function with null Euler characteristic, 2019.**