Tree decompositions for probabilistic data management

Pierre Senellart
(joint work with Antoine Amarilli, Pierre Bourhis, Silviu Maniu, Mikaël Monet)

2 May 2017

IRIF, Algorithms & complexity seminar

Slides due in large part to Antoine Amarilli

Query evaluation

- Relational signature σ
  - Example: R (arity 1), S (arity 2), T (arity 1)

Query evaluation

- Relational signature σ
  - Example: R (arity 1), S (arity 2), T (arity 1)
- Class $\mathcal{I}$ of instances
  - Example: all instances; acyclic instances; treelike instances; ...

Query evaluation

- Relational signature σ
  - Example: R (arity 1), S (arity 2), T (arity 1)
- Class $\mathcal{I}$ of instances
  - Example: all instances; acyclic instances; treelike instances; ...
- Fragment $\mathcal{Q}$ of Boolean constant-free queries
  - Example: Boolean conjunctive queries (= existentially quantified conjunction of atoms)
  - Example of CQ: $q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$

Query evaluation

- Relational signature σ
  - Example: R (arity 1), S (arity 2), T (arity 1)
- Class $\mathcal{I}$ of instances
  - Example: all instances; acyclic instances; treelike instances; ...
- Fragment $\mathcal{Q}$ of Boolean constant-free queries
  - Example: Boolean conjunctive queries (= existentially quantified conjunction of atoms)
  - Example of CQ: $q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$
- Query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Fix a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$
  - Determine whether I satisfies q (written $I \models q$)
  - Data complexity: Complexity as a function of I, not q
  - Combined complexity: Complexity as a function of both I and q

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R
a
b
c

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R
a
b
c

S	
a	a
b	v
b	w

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	S		T
a	a	a	v
b	b	v	w
c	b	w	b

Query evaluation example

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	S		T
a	a	a	v
b	b	v	w
c	b	w	b

Probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of queries, class $\mathcal{I}$ of instances.

- Probabilistic query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Given a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$

Probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of queries, class $\mathcal{I}$ of instances.

- Probabilistic query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Given a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$
  - And given a probability valuation π mapping facts of I to probabilities in $[0, 1]$

Probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of queries, class $\mathcal{I}$ of instances.

- Probabilistic query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Given a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$
  - And given a probability valuation π mapping facts of I to probabilities in $[0, 1]$
  - Compute the probability that $I \models q$

Probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of queries, class $\mathcal{I}$ of instances.

- Probabilistic query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Given a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$
  - And given a probability valuation π mapping facts of I to probabilities in $[0, 1]$
  - Compute the probability that $I \models q$
  - Data complexity: measured as a function of I and π
  - Combined complexity: measured as a function of q, I, and π

Probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of queries, class $\mathcal{I}$ of instances.

- Probabilistic query evaluation problem for $\mathcal{Q}$ and $\mathcal{I}$:
  - Given a query $q \in \mathcal{Q}$
  - Given an input instance $I \in \mathcal{I}$
  - And given a probability valuation π mapping facts of I to probabilities in $[0, 1]$
  - Compute the probability that $I \models q$
  - Data complexity: measured as a function of I and π
  - Combined complexity: measured as a function of q, I, and π

- Semantics: (I, π) gives a probability distribution on $I' \subseteq I$:
  - Each fact $F \in I$ is either present or absent with probability $\pi(F)$
  - Facts are independent

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

This (I, π) represents the following probability distribution:

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

This (I, π) represents the following probability distribution:

$.5 \times .2$
S
a a
b v
b w

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

This (I, π) represents the following probability distribution:

$.5 \times .2$	
S	
a	a
b	v
b	w

$.5 \times (1 - .2)$	
S	
a	a
b	v

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

This (I, π) represents the following probability distribution:

$.5 \times .2$	
S	
a	a
b	v
b	w

$.5 \times (1 - .2)$	
S	
a	a
b	v

$(1 - .5) \times .2$	
S	
a	a
b	w

Example of a probabilistic instance

S		
a	a	1
b	v	.5
b	w	.2

This (I, π) represents the following probability distribution:

$.5 \times .2$	
S	
a	a
b	v
b	w

$.5 \times (1 - .2)$	
S	
a	a
b	v

$(1 - .5) \times .2$	
S	
a	a
b	w

$(1 - .5) \times (1 - .2)$	
S	
a	a

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability:

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability: .4 $\times$

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability: $.4 \times (1 -$

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability: $.4 \times (1 - (1 - .5 \times .3)$

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability: $.4 \times (1 - (1 - .5 \times .3) \times (1 - .2 \times .7))$

Example of probabilistic query evaluation

Signature σ, class $\mathcal{Q}$ of conjunctive queries, class $\mathcal{I}$ of all instances.

$$q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$$

R	
a	1
b	.4
c	.6

S		
a	a	1
b	v	.5
b	w	.2

T	
v	.3
w	.7
b	1

- The query is true iff $R(b)$ is here and one of:
  - $S(b, v)$ and $T(v)$ are here
  - $S(b, w)$ and $T(w)$ are here
- Probability: $.4 \times (1 - (1 - .5 \times .3) \times (1 - .2 \times .7)) = .1076$

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries
  - PQE is PTIME for any $q \in \mathcal{S}$ on all instances

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries
  - PQE is PTIME for any $q \in \mathcal{S}$ on all instances
  - PQE is #P-hard for any $q \in \mathcal{Q} \backslash \mathcal{S}$ on all instances

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries
  - PQE is PTIME for any $q \in \mathcal{S}$ on all instances
  - PQE is #P-hard for any $q \in \mathcal{Q} \backslash \mathcal{S}$ on all instances
  - $q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$ is unsafe!

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries
  - PQE is PTIME for any $q \in \mathcal{S}$ on all instances
  - PQE is #P-hard for any $q \in \mathcal{Q} \backslash \mathcal{S}$ on all instances
  - $q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$ is unsafe!

Is there a smaller class $\mathcal{I}$ such that
PQE is tractable for a larger $\mathcal{Q}$?

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) complexity of probabilistic query evaluation depending on the class $\mathcal{Q}$ of queries and class $\mathcal{I}$ of instances?

- Existing dichotomy result: (Dalvi and Suciu, 2012)
  - $\mathcal{Q}$ are (unions of) conjunctive queries, $\mathcal{I}$ is all instances
  - There is a class $\mathcal{S} \subseteq \mathcal{Q}$ of safe queries
  - PQE is PTIME for any $q \in \mathcal{S}$ on all instances
  - PQE is #P-hard for any $q \in \mathcal{Q} \backslash \mathcal{S}$ on all instances
  - $q : \exists x\, y\ R(x) \wedge S(x, y) \wedge T(y)$ is unsafe!

Is there a smaller class $\mathcal{I}$ such that
PQE is tractable for a larger $\mathcal{Q}$?

- Probabilistic XML: (Cohen et al., 2009)
  - $\mathcal{Q}$ are deterministic tree automata, $\mathcal{I}$ are trees
  - PQE is PTIME in combined complexity

Trees and treelike instances

- Goal: find an instance class $\mathcal{I}$ where PQE is tractable

Trees and treelike instances

- Goal: find an instance class $\mathcal{I}$ where PQE is tractable
- Idea: take $\mathcal{I}$ to be treelike instances
  - Treelike: the treewidth is bounded by a constant

Trees and treelike instances

- Goal: find an instance class $\mathcal{I}$ where PQE is tractable
- Idea: take $\mathcal{I}$ to be treelike instances
  - Treelike: the treewidth is bounded by a constant
    - Trees have treewidth 1
    - Cycles have treewidth 2
    - k-cliques and $(k-1)$-grids have treewidth $k-1$

Trees and treelike instances

- Goal: find an instance class $\mathcal{I}$ where PQE is tractable
- Idea: take $\mathcal{I}$ to be treelike instances
  - Treelike: the treewidth is bounded by a constant
    - Trees have treewidth 1
    - Cycles have treewidth 2
    - k-cliques and $(k-1)$-grids have treewidth $k-1$
- For non-probabilistic query evaluation (Courcelle, 1990):
  - $\mathcal{I}$: treelike instances; $\mathcal{Q}$: monadic second-order (MSO) queries
  - non-probabilistic QE is in linear time (in data complexity)

Trees and treelike instances

- Goal: find an instance class $\mathcal{I}$ where PQE is tractable
- Idea: take $\mathcal{I}$ to be treelike instances
  - Treelike: the treewidth is bounded by a constant
    - Trees have treewidth 1
    - Cycles have treewidth 2
    - k-cliques and $(k-1)$-grids have treewidth $k-1$
- For non-probabilistic query evaluation (Courcelle, 1990):
  - $\mathcal{I}$: treelike instances; $\mathcal{Q}$: monadic second-order (MSO) queries
  - non-probabilistic QE is in linear time (in data complexity)
- Does this extend to probabilistic QE?

Main result (Amarilli et al., 2015, 2016)

An instance-based dichotomy result on data complexity:

Upper bound. For $\mathcal{I}$ the treelike instances and $\mathcal{Q}$ the MSO queries

- PQE is in linear time modulo arithmetic costs

Main result (Amarilli et al., 2015, 2016)

An instance-based dichotomy result on data complexity:

Upper bound. For $\mathcal{I}$ the treelike instances and $\mathcal{Q}$ the MSO queries

- PQE is in linear time modulo arithmetic costs

Lower bound. For any unbounded-tw family $\mathcal{I}$ and $\mathcal{Q}$ FO queries

- PQE is #P-hard under RP reductions assuming
  - Signature arity is 2 (graphs)
  - High-tw instances in $\mathcal{I}$ are easily constructible

Outline

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

- Provenance: $(f_1 \wedge f_2)$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

- Provenance: $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

- Provenance: $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

- Provenance: $(f_1 \wedge f_2) \vee (f_3 \wedge f_4) \vee f_5$

Technical tool: provenance

The provenance of a query q on an instance I:

- Boolean function φ whose variables are the facts of I
- A subinstance of I satisfies q iff φ is true for that valuation
- For all $\nu : I \to \{0, 1\}$ we have $\nu(\varphi) = 1$ iff $\{F \in I \mid \nu(F) = 1\} \models q$

Example query: $\exists x\, y\, z\ R(x, y) \wedge R(y, z)$

R		
a	b	f_1
b	c	f_2
d	e	f_3
e	d	f_4
f	f	f_5

- Provenance: $(f_1 \wedge f_2) \vee (f_3 \wedge f_4) \vee f_5$

General roadmap

- Use provenance for probabilistic query evaluation:
  - Compute a provenance representation efficiently
  - Probability of the provenance = probability of the query

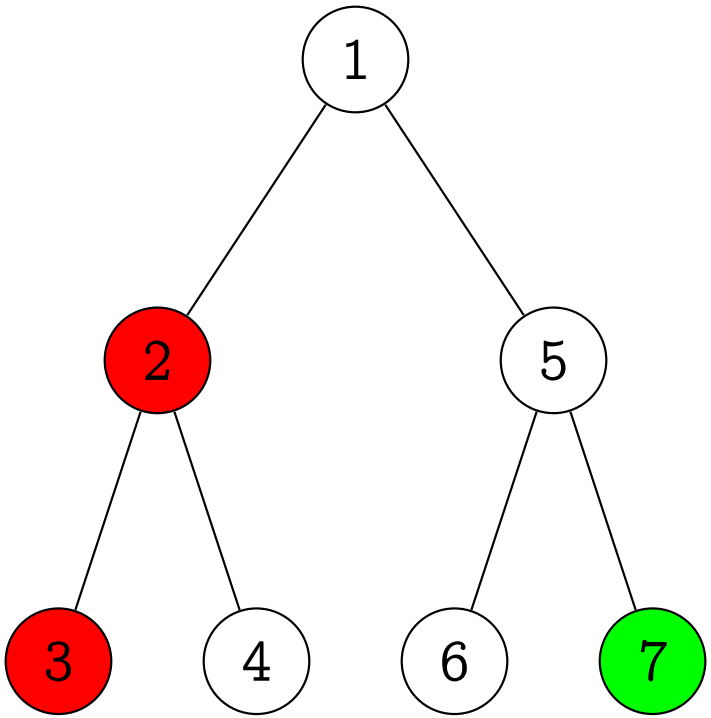
General roadmap

- Use provenance for probabilistic query evaluation:
  - Compute a provenance representation efficiently
  - Probability of the provenance = probability of the query
  - Compute the provenance probability efficiently
    (show it is not #P-hard as in the general case)

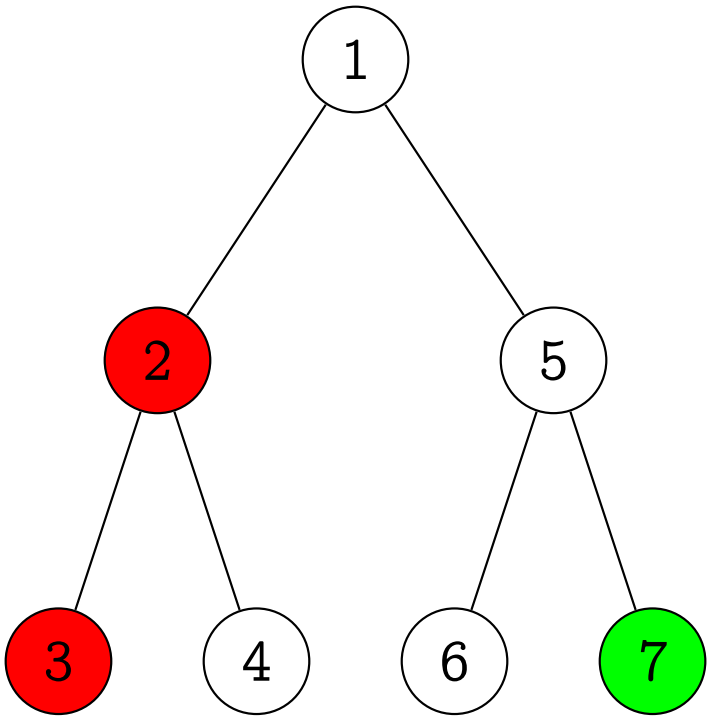
General roadmap

- Use provenance for probabilistic query evaluation:
  - Compute a provenance representation efficiently
  - Probability of the provenance = probability of the query
  - Compute the provenance probability efficiently
    (show it is not #P-hard as in the general case)
- To solve the PQE problem on treelike instances for MSO
  - First solve the problem on trees with tree automata
  - Then use the results of (Courcelle, 1990)

Uncertain trees

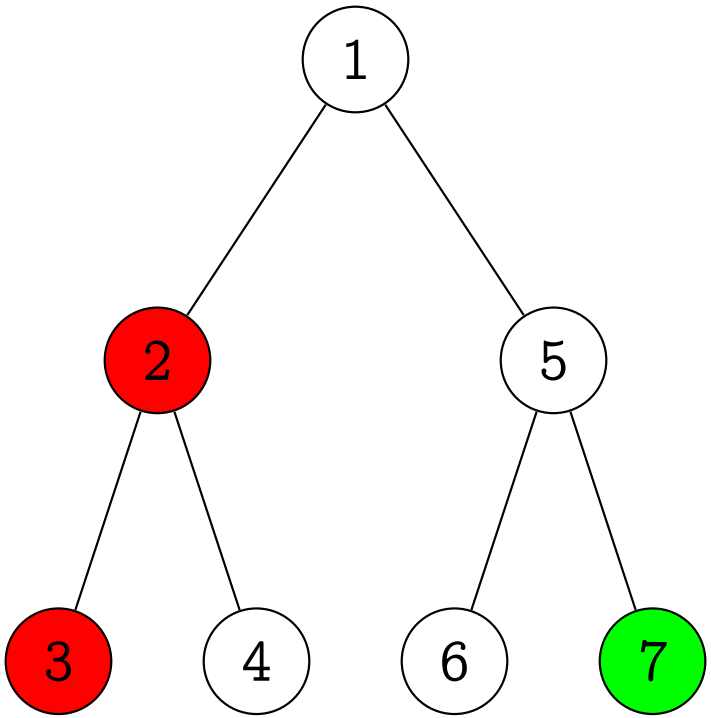


Uncertain trees



A valuation of a tree decides whether to keep or discard node labels.

Uncertain trees



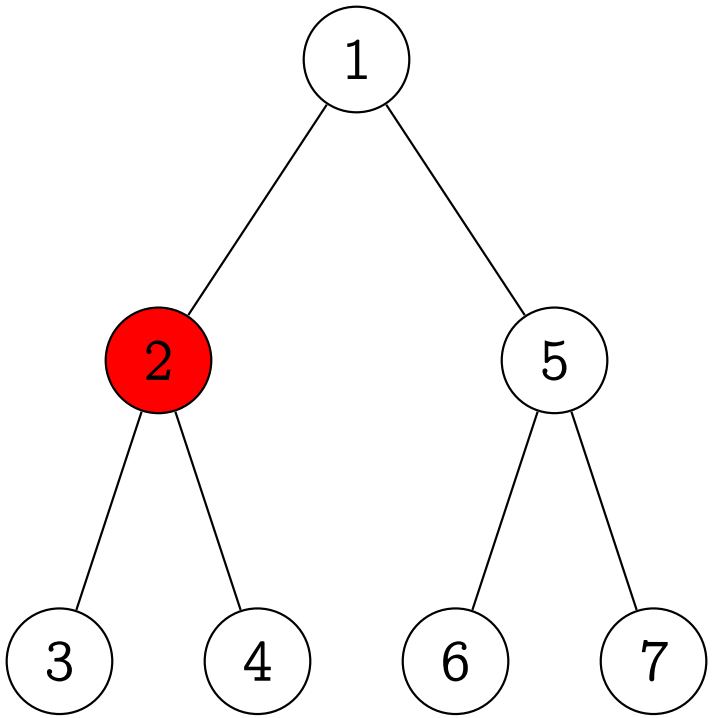
A valuation of a tree decides whether to keep or discard node labels.

Example tree automaton:
"Is there both a red and a green node?"

Valuation: $\{2, 3, 7\}$

The tree automaton accepts

Uncertain trees



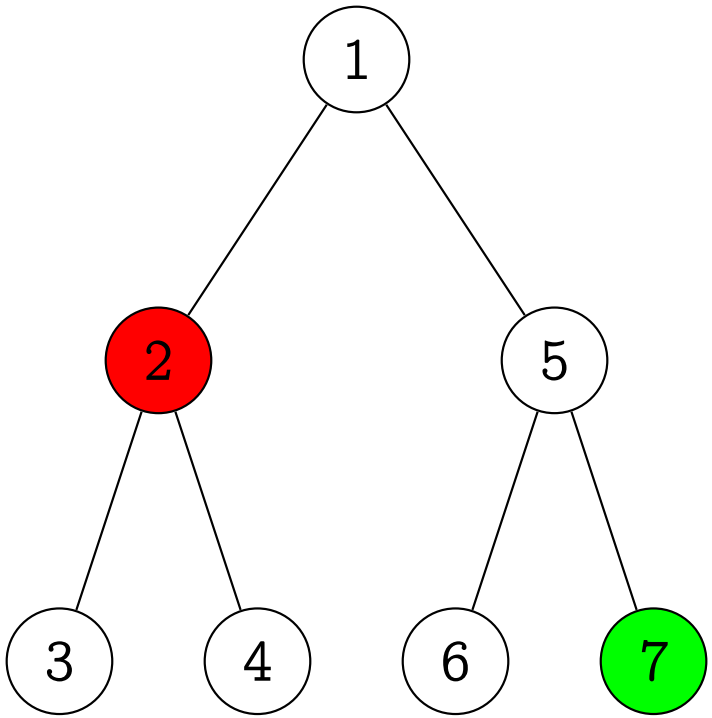
A valuation of a tree decides whether to keep or discard node labels.

Example tree automaton:
"Is there both a red and a green node?"

Valuation: $\{2\}$

The tree automaton rejects

Uncertain trees



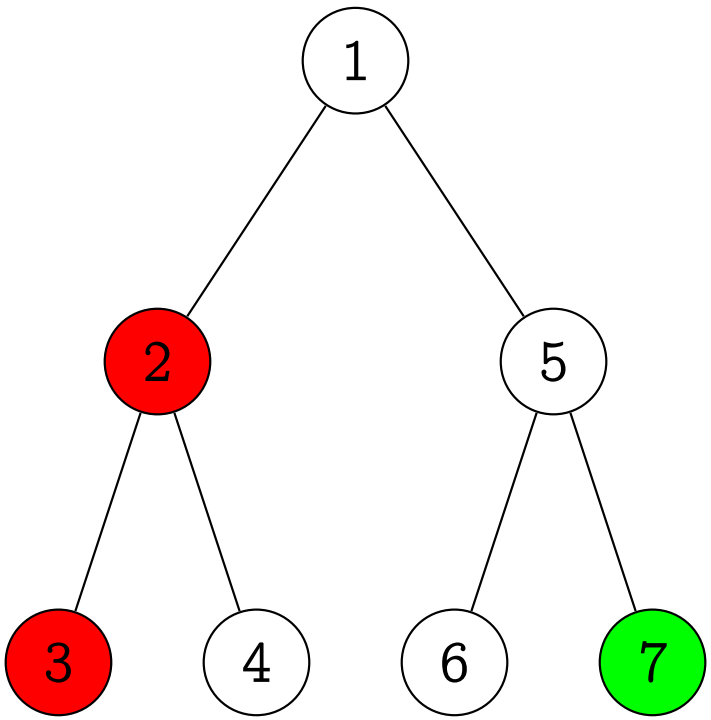
A valuation of a tree decides whether to keep or discard node labels.

Example tree automaton:
"Is there both a red and a green node?"

Valuation: $\{2, 7\}$

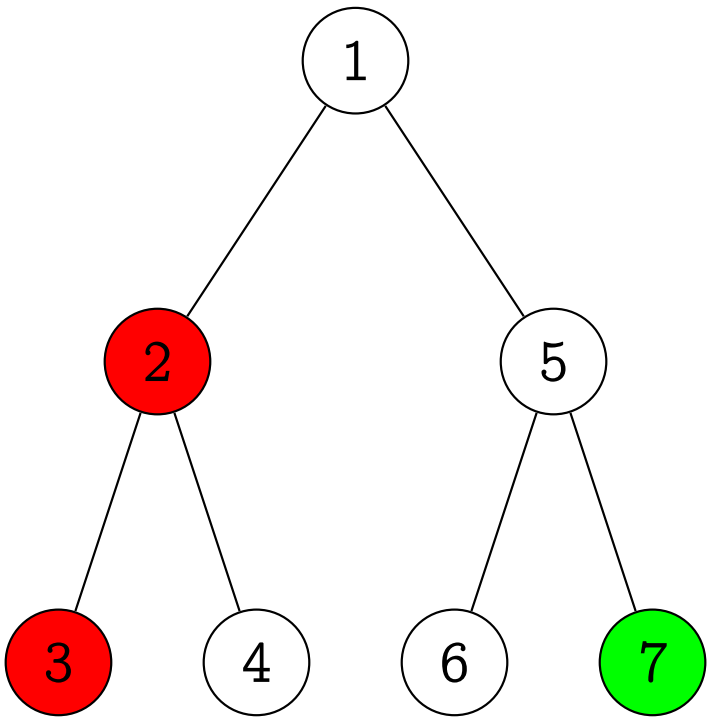
The tree automaton accepts

Provenance formulae and circuits on trees



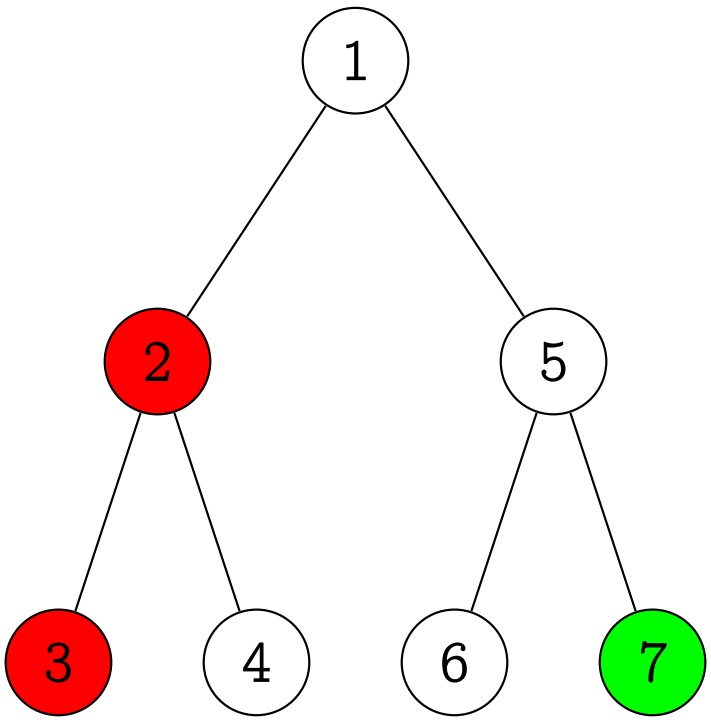
- Which valuations satisfy the query?

Provenance formulae and circuits on trees



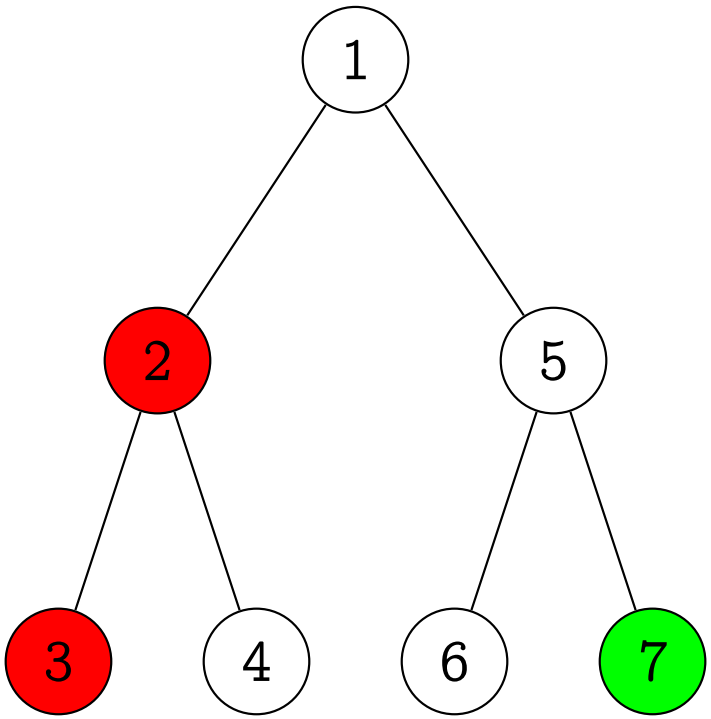
- Which valuations satisfy the query?
- Provenance of a tree automaton A on an uncertain tree T:
  - Boolean function φ
  - on variables x_2, x_3, x_7
  - A accepts $\nu(T)$ iff $\nu(\varphi)$ is true

Provenance formulae and circuits on trees



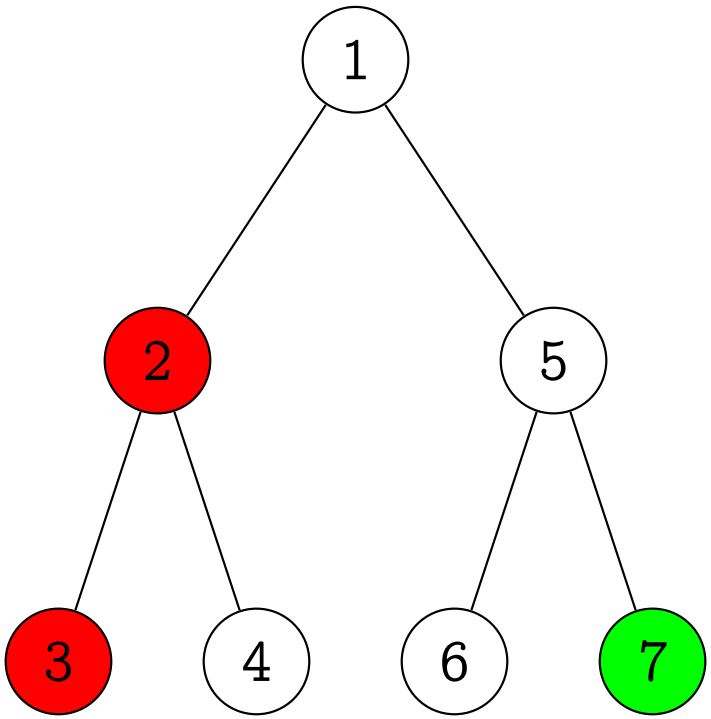
- Which valuations satisfy the query?
- Provenance of a tree automaton A on an uncertain tree T:
  - Boolean function φ
  - on variables x_2, x_3, x_7
  - A accepts $\nu(T)$ iff $\nu(\varphi)$ is true
- Provenance circuit of A on T (Deutch et al., 2014)
  - Boolean circuit C
  - with input gates g_2, g_3, g_7
  - A accepts $\nu(T)$ iff $\nu(C)$ is true

Example



Is there both a red and a green node?

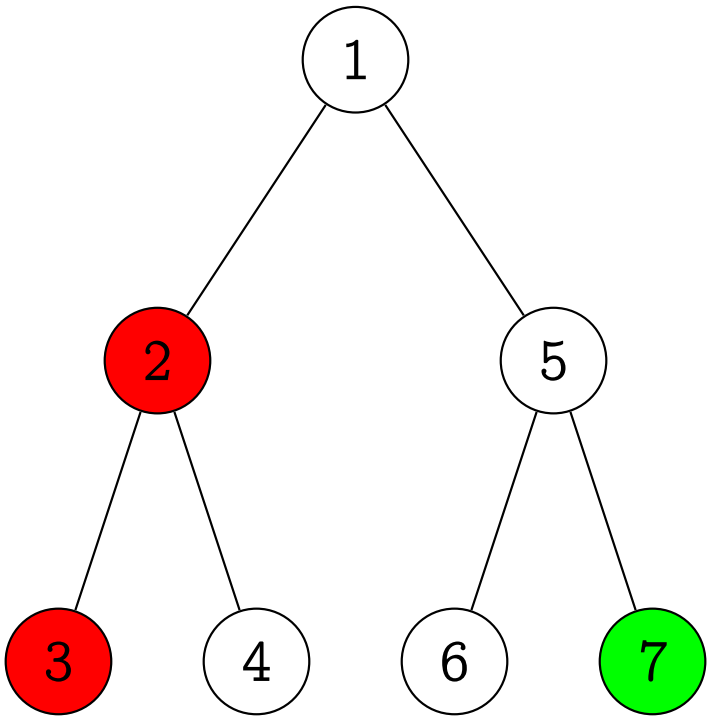
Example



Is there both a red and a green node?

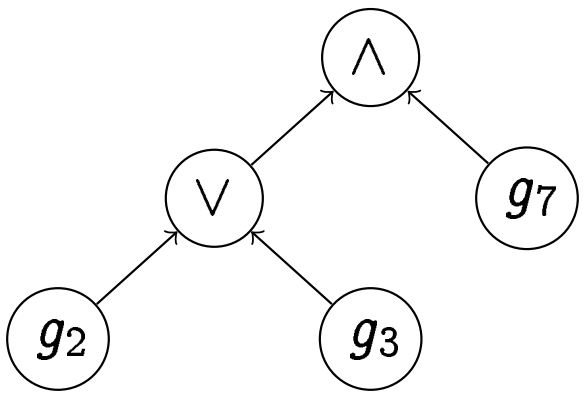
- Provenance formula:
  $(x_2 \vee x_3) \wedge x_7$

Example



Is there both a red and a green node?

- Provenance formula:
  $(x_2 \vee x_3) \wedge x_7$
- Provenance circuit:



Upper bound on trees

Theorem (Amarilli et al. (2015))

For any bottom-up (nondeterministic) tree automaton A and input tree T, we can build a provenance circuit of A on T in linear time in A and T.

Upper bound on trees

Theorem (Amarilli et al. (2015))

For any bottom-up (nondeterministic) tree automaton A and input tree T, we can build a provenance circuit of A on T in linear time in A and T.

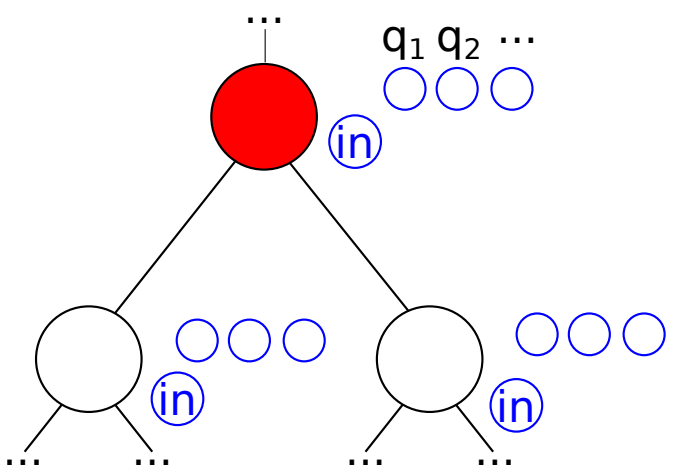
Construct the Boolean provenance circuit bottom-up

Upper bound on trees

Theorem (Amarilli et al. (2015))

For any bottom-up (nondeterministic) tree automaton A and input tree T, we can build a provenance circuit of A on T in linear time in A and T.

Construct the Boolean provenance circuit bottom-up

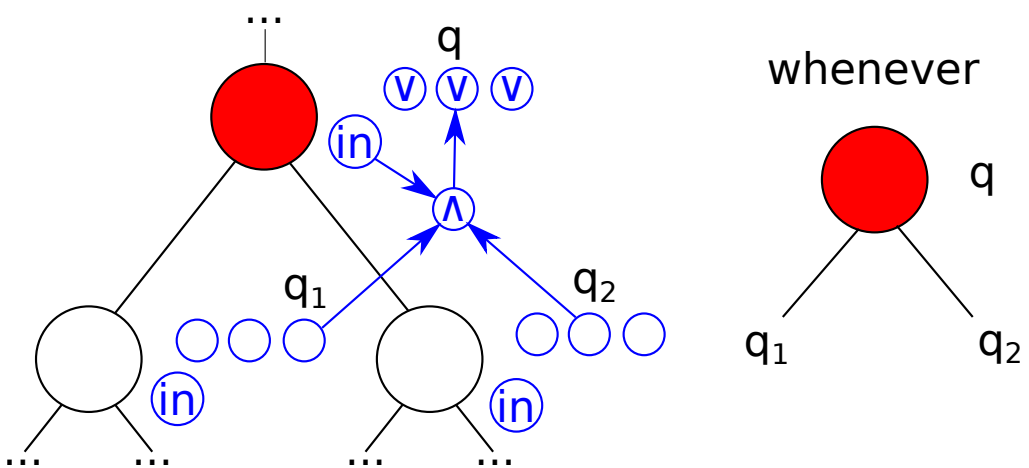


Upper bound on trees

Theorem (Amarilli et al. (2015))

For any bottom-up (nondeterministic) tree automaton A and input tree T, we can build a provenance circuit of A on T in linear time in A and T.

Construct the Boolean provenance circuit bottom-up

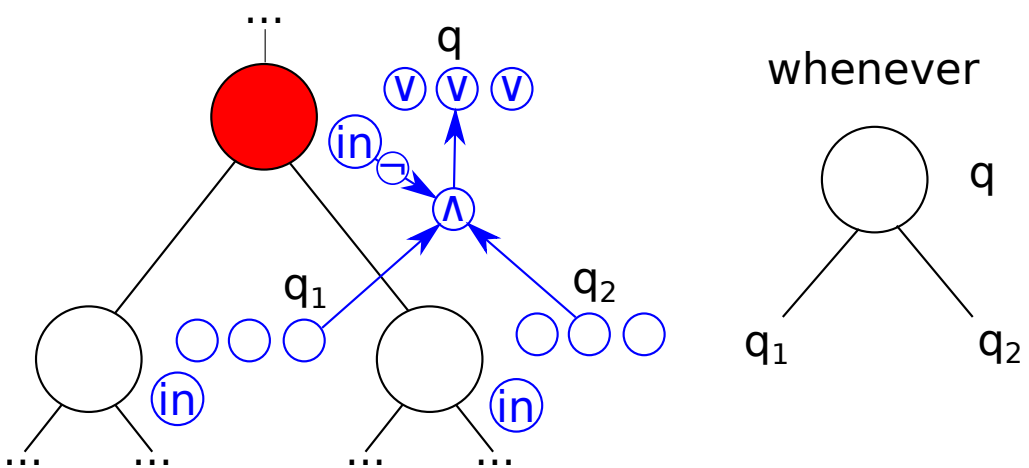


Upper bound on trees

Theorem (Amarilli et al. (2015))

For any bottom-up (nondeterministic) tree automaton A and input tree T, we can build a provenance circuit of A on T in linear time in A and T.

Construct the Boolean provenance circuit bottom-up



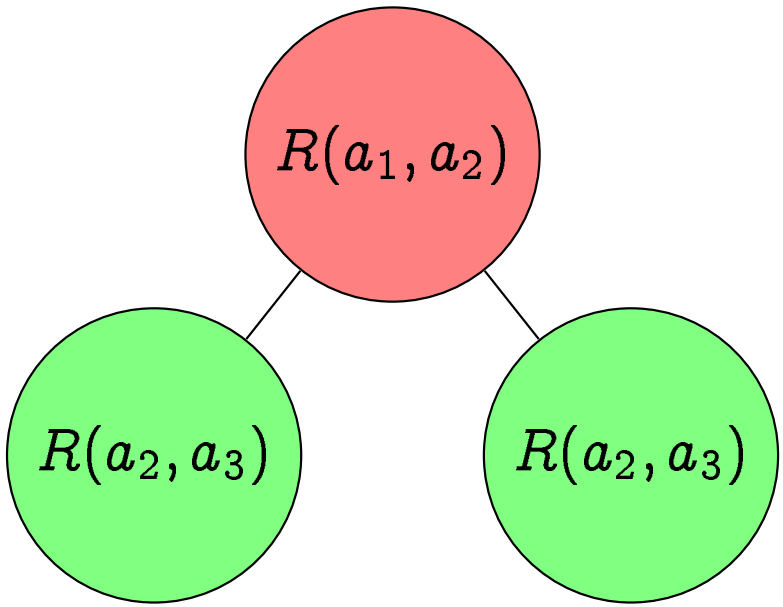
Treelike instances

- Treelike instance I
- Tree encoding: tree E on fixed alphabet, represents I
- MSO query on I translates to
  - $\rightarrow$ MSO query on E by (Courcelle, 1990)
  - $\rightarrow$ tree automaton on E by (Thatcher and Wright, 1968)

Treelike instances

- Treelike instance I
- Tree encoding: tree E on fixed alphabet, represents I
- MSO query on I translates to
  - $\rightarrow$ MSO query on E by (Courcelle, 1990)
  - $\rightarrow$ tree automaton on E by (Thatcher and Wright, 1968)
- Uncertain instance: each fact can be present or absent
- Possible subinstances are possible valuations of the encoding

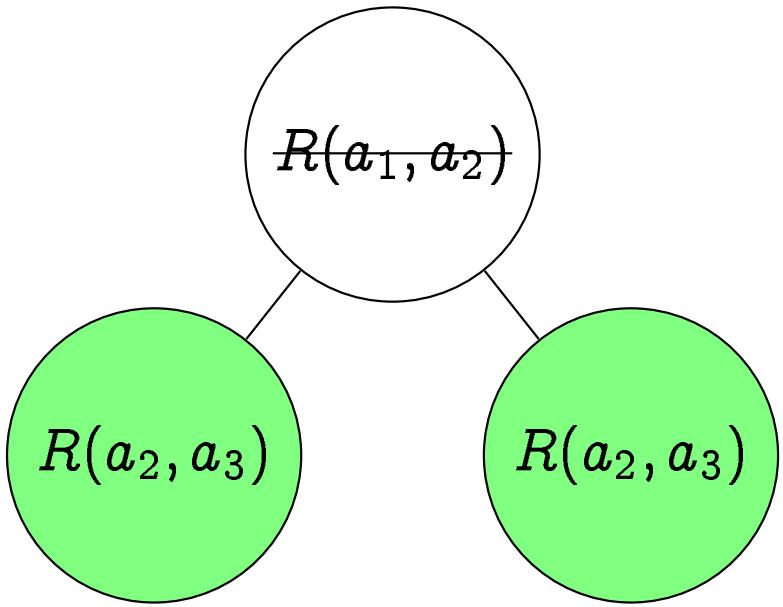
R	
a	b
b	c
b	d



Treelike instances

- Treelike instance I
- Tree encoding: tree E on fixed alphabet, represents I
- MSO query on I translates to
  - $\rightarrow$ MSO query on E by (Courcelle, 1990)
  - $\rightarrow$ tree automaton on E by (Thatcher and Wright, 1968)
- Uncertain instance: each fact can be present or absent
- Possible subinstances are possible valuations of the encoding

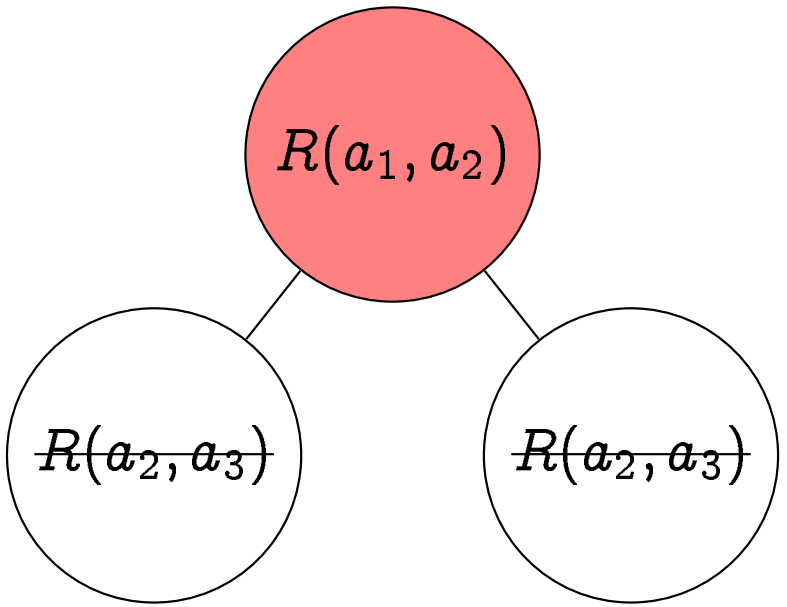
R	
~~a~~	~~b~~
b	c
b	d



Treelike instances

- Treelike instance I
- Tree encoding: tree E on fixed alphabet, represents I
- MSO query on I translates to
  - $\rightarrow$ MSO query on E by (Courcelle, 1990)
  - $\rightarrow$ tree automaton on E by (Thatcher and Wright, 1968)
- Uncertain instance: each fact can be present or absent
- Possible subinstances are possible valuations of the encoding

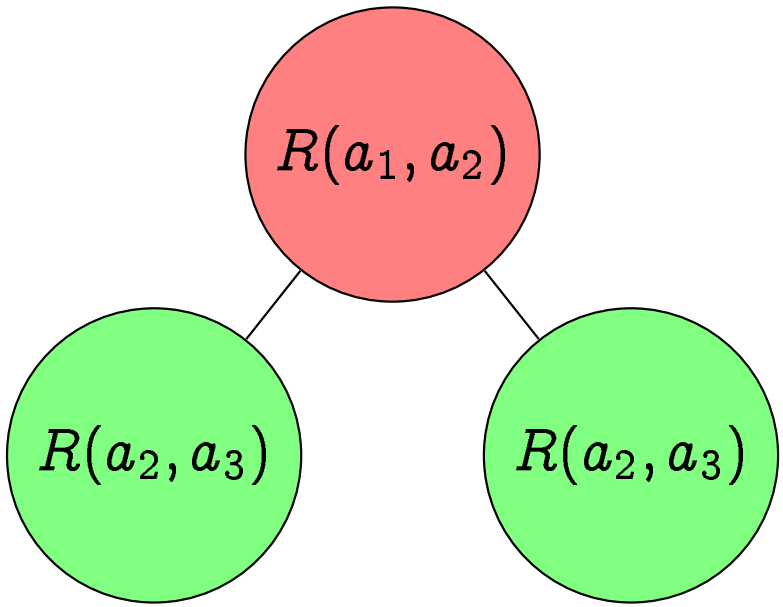
R	
a	b
~~b~~	~~c~~
~~b~~	~~d~~



Treelike instances

- Treelike instance I
- Tree encoding: tree E on fixed alphabet, represents I
- MSO query on I translates to
  - $\rightarrow$ MSO query on E by (Courcelle, 1990)
  - $\rightarrow$ tree automaton on E by (Thatcher and Wright, 1968)
- Uncertain instance: each fact can be present or absent
- Possible subinstances are possible valuations of the encoding

R	
a	b
b	c
b	d



Upper bound on treelike instances

Theorem (Amarilli et al. (2015))

For any fixed MSO query q and $k \in \mathbb{N}$, for any input instance I of treewidth $\leq k$, we can build in linear time in I a provenance circuit of q on I.

Probability evaluation

Two alternate ways to see why probability evaluation is tractable on our provenance circuits:

Probability evaluation

Two alternate ways to see why probability evaluation is tractable on our provenance circuits:

- They have bounded treewidth themselves
  - Follows the structure of the tree encoding
  - Width only depends on number of automaton states
  - Apply message passing (Lauritzen and Spiegelhalter, 1988)

Probability evaluation

Two alternate ways to see why probability evaluation is tractable on our provenance circuits:

- They have bounded treewidth themselves
  - Follows the structure of the tree encoding
  - Width only depends on number of automaton states
  - Apply message passing (Lauritzen and Spiegelhalter, 1988)
- If the tree automaton is deterministic
  - All conjunctions depend on disjoint sets of input gates
  - All disjunctions are on mutually exclusive outcomes
  - Circuit is a d-DNNF (Darwiche, 2001)

Probability evaluation

Two alternate ways to see why probability evaluation is tractable on our provenance circuits:

- They have bounded treewidth themselves
  - Follows the structure of the tree encoding
  - Width only depends on number of automaton states
  - Apply message passing (Lauritzen and Spiegelhalter, 1988)
- If the tree automaton is deterministic
  - All conjunctions depend on disjoint sets of input gates
  - All disjunctions are on mutually exclusive outcomes
  - Circuit is a d-DNNF (Darwiche, 2001)

Corollary (Amarilli et al. (2015))

Probabilistic query evaluation of MSO queries on treelike instances is in linear time in data complexity up to arithmetic operations.

What about combined complexity?

- Simple limitation: QE is NP-hard in combined complexity when $\mathcal{Q}$ is conjunctive queries and $\mathcal{I}$ is a fixed instance

What about combined complexity?

- Simple limitation: QE is NP-hard in combined complexity when $\mathcal{Q}$ is conjunctive queries and $\mathcal{I}$ is a fixed instance
- $\Rightarrow$ We need to restrict the query language as well

What about combined complexity?

- Simple limitation: QE is NP-hard in combined complexity when $\mathcal{Q}$ is conjunctive queries and $\mathcal{I}$ is a fixed instance
- $\Rightarrow$ We need to restrict the query language as well
- Possible restriction: ICG-Datalog of bounded body size (Amarilli et al., 2017a)
  - recursive query language with mechanism to build new facts (Datalog)
  - with limited use of negation (stratified Datalog)
  - with guards on the use of variables in new facts
  - with a bound on the size of rules producing new facts

What about combined complexity?

- Simple limitation: QE is NP-hard in combined complexity when $\mathcal{Q}$ is conjunctive queries and $\mathcal{I}$ is a fixed instance
- $\Rightarrow$ We need to restrict the query language as well
- Possible restriction: ICG-Datalog of bounded body size (Amarilli et al., 2017a)
  - recursive query language with mechanism to build new facts (Datalog)
  - with limited use of negation (stratified Datalog)
  - with guards on the use of variables in new facts
  - with a bound on the size of rules producing new facts
- Technical tool: cycluits, or cyclic circuits

What about combined complexity?

- Simple limitation: QE is NP-hard in combined complexity when $\mathcal{Q}$ is conjunctive queries and $\mathcal{I}$ is a fixed instance
- $\Rightarrow$ We need to restrict the query language as well
- Possible restriction: ICG-Datalog of bounded body size (Amarilli et al., 2017a)
  - recursive query language with mechanism to build new facts (Datalog)
  - with limited use of negation (stratified Datalog)
  - with guards on the use of variables in new facts
  - with a bound on the size of rules producing new facts
- Technical tool: cycluits, or cyclic circuits

Theorem (Amarilli et al. (2017a))

Given an ICG-Datalog program q of body size k_q and a relational instance I of treewidth k_I, we can compute in time $O(f(k_I, k_q) \cdot |I| \cdot |q|)$ a provenance cycluit of q on I.

Combined complexity of PQE (1/2)

- Cycluits can be converted into circuits in polynomial-time (Riedel and Bruck, 2012)...

Combined complexity of PQE (1/2)

- Cycluits can be converted into circuits in polynomial-time (Riedel and Bruck, 2012)...
- but this translation does not preserve the treewidth, which prevents obtaining a polynomial-time upper bound this way

Combined complexity of PQE (1/2)

- Cycluits can be converted into circuits in polynomial-time (Riedel and Bruck, 2012)...
- but this translation does not preserve the treewidth, which prevents obtaining a polynomial-time upper bound this way
- We can show a 2EXPTIME upper bound on the combined complexity of PQE (Amarilli et al., 2017a) of ICG-Datalog programs of bounded body size on treelike instance (better than the naive non-elementary bound!)

Combined complexity of PQE (2/2)

- We cannot do much better (Amarilli et al., 2017b). We obtain intractability in combined complexity (#P-hardness):
  - when the instance is a tree, and the query is a simple directed path
  - when the instance is a directed tree, and the query is a path or a downwards tree

Combined complexity of PQE (2/2)

- We cannot do much better (Amarilli et al., 2017b). We obtain intractability in combined complexity (#P-hardness):
  - when the instance is a tree, and the query is a simple directed path
  - when the instance is a directed tree, and the query is a path or a downwards tree
- Very limited settings of combined tractability of PQE
  - when the instance is a path, and the query is a connected query
  - when the instance is a directed tree, and the query is a directed path

Outline

Lower bound goal

- Class $\mathcal{I}$ of unbounded-treewidth instances, query q in class $\mathcal{Q}$.
- Show that probabilistic query evaluation of q on $\mathcal{I}$ is hard

Lower bound goal

- Class $\mathcal{I}$ of unbounded-treewidth instances, query q in class $\mathcal{Q}$.
- Show that probabilistic query evaluation of q on $\mathcal{I}$ is hard
- Restrict to arity-2 (= labeled graphs) for technical reasons

Lower bound goal

- Class $\mathcal{I}$ of unbounded-treewidth instances, query q in class $\mathcal{Q}$.
- Show that probabilistic query evaluation of q on $\mathcal{I}$ is hard
- Restrict to arity-2 (= labeled graphs) for technical reasons
- Impose that $\mathcal{I}$ is tw-constructible:

Lower bound goal

- Class $\mathcal{I}$ of unbounded-treewidth instances, query q in class $\mathcal{Q}$.
- Show that probabilistic query evaluation of q on $\mathcal{I}$ is hard
- Restrict to arity-2 (= labeled graphs) for technical reasons
- Impose that $\mathcal{I}$ is tw-constructible:
  - Given $k \in \mathbb{N}$, we can construct in time $\mathrm{Poly}(k)$ an instance of $\mathcal{I}$ of treewidth $\geq k$

Lower bound goal

- Class $\mathcal{I}$ of unbounded-treewidth instances, query q in class $\mathcal{Q}$.
- Show that probabilistic query evaluation of q on $\mathcal{I}$ is hard
- Restrict to arity-2 (= labeled graphs) for technical reasons
- Impose that $\mathcal{I}$ is tw-constructible:
  - Given $k \in \mathbb{N}$, we can construct in time Poly(k) an instance of $\mathcal{I}$ of treewidth $\geq k$
  - Otherwise instances of treewidth k in $\mathcal{I}$ could be very large... see (Makowsky and Marino, 2003)

Our lower bound result

Theorem (Amarilli et al. (2016))

There is a first-order query q such that for any unbounded-tw, tw-constructible, arity-2 instance family $\mathcal{I}$, PQE for q on $\mathcal{I}$ is #P-hard under RP reductions.

Idea: extracting topological minors

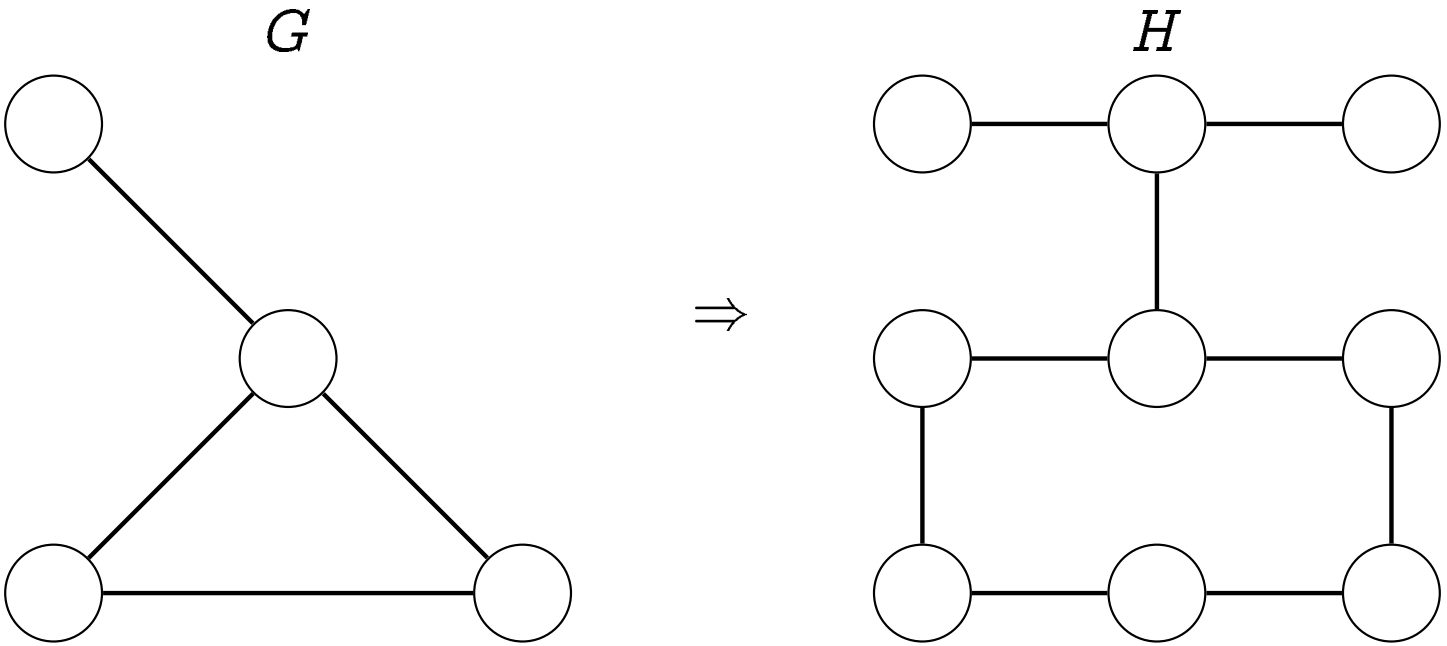
- Let G be a planar graph of degree ≤ 3

Idea: extracting topological minors

- Let G be a planar graph of degree ≤ 3
- G is a topological minor of H if:

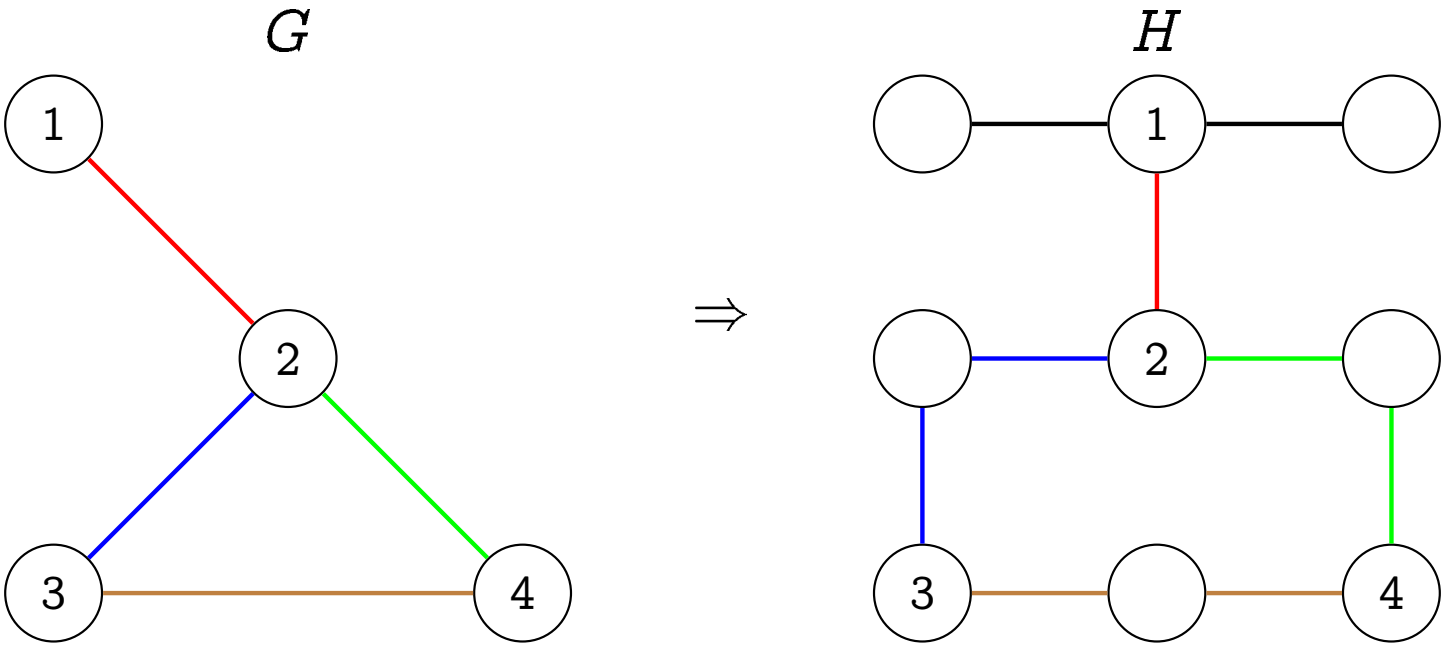
Idea: extracting topological minors

- Let G be a planar graph of degree ≤ 3
- G is a topological minor of H if:



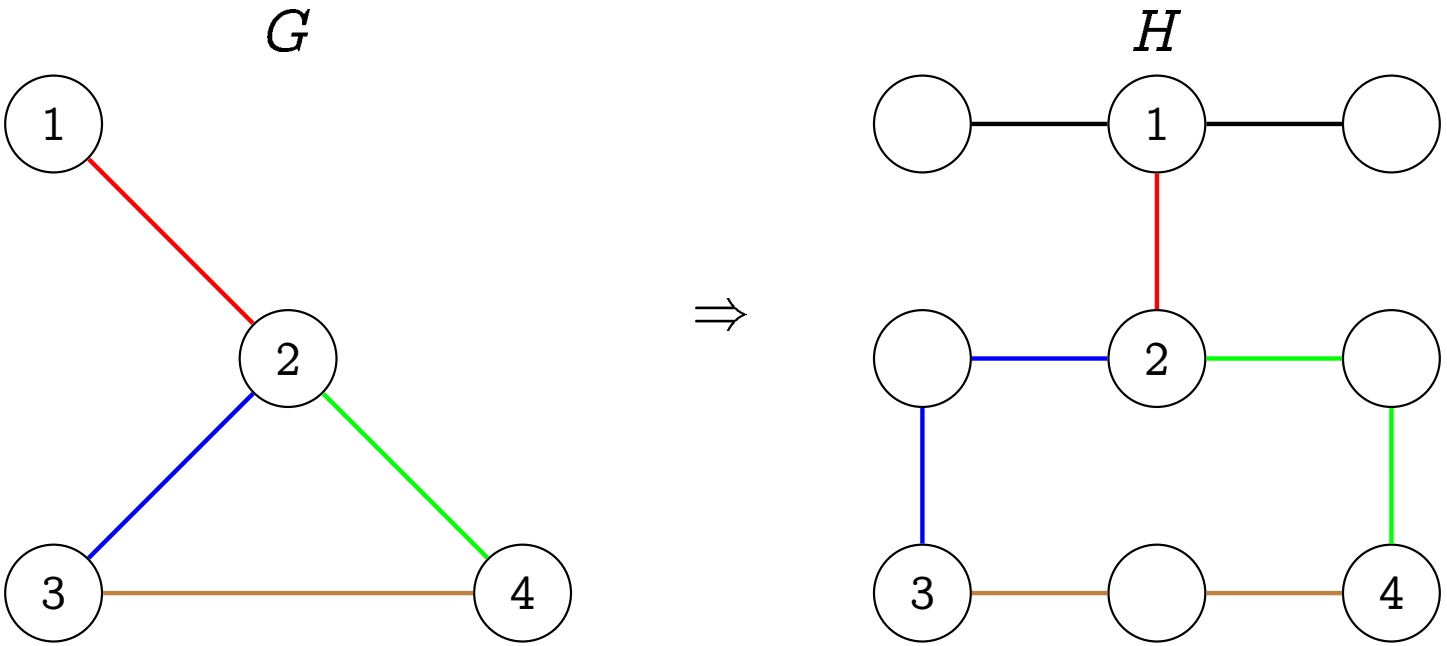
Idea: extracting topological minors

- Let G be a planar graph of degree ≤ 3
- G is a topological minor of H if:



Idea: extracting topological minors

- Let G be a planar graph of degree ≤ 3
- G is a topological minor of H if:



- Map vertices to vertices
- Map edges to vertex-disjoint paths

Topological minor extraction results

Theorem (Robertson and Seymour (1986))

For any planar graph G of degree ≤ 3,
for any graph H of sufficiently high treewidth,
G is a topological minor of H.

Topological minor extraction results

Theorem (Robertson and Seymour (1986))

For any planar graph G of degree ≤ 3,
for any graph H of sufficiently high treewidth,
G is a topological minor of H.

More recently:

Theorem (Chekuri and Chuzhoy (2014))

There is a certain constant $c \in \mathbb{N}$ such that
for any planar graph G of degree ≤ 3,
for any graph H of treewidth $\geq |G|^c$,
G is a topological minor of H and
we can embed G in H (w.h.p.) in PTIME in $|H|$.

Intuition for our result: reduction

- Choose a problem from which to reduce:
  - Must be #P-hard on planar degree-3 graphs
  - Must be encodable to an FO query q (more later)
  - We use the problem of counting matchings

Intuition for our result: reduction

- Choose a problem from which to reduce:
  - Must be #P-hard on planar degree-3 graphs
  - Must be encodable to an FO query q (more later)
  - We use the problem of counting matchings
- Given an input graph G, compute $k := |G|^c$

Intuition for our result: reduction

- Choose a problem from which to reduce:
  - Must be #P-hard on planar degree-3 graphs
  - Must be encodable to an FO query q (more later)
  - We use the problem of counting matchings
- Given an input graph G, compute $k := |G|^c$
- Compute in PTIME an instance I of $\mathcal{I}$ of treewidth $\geq k$

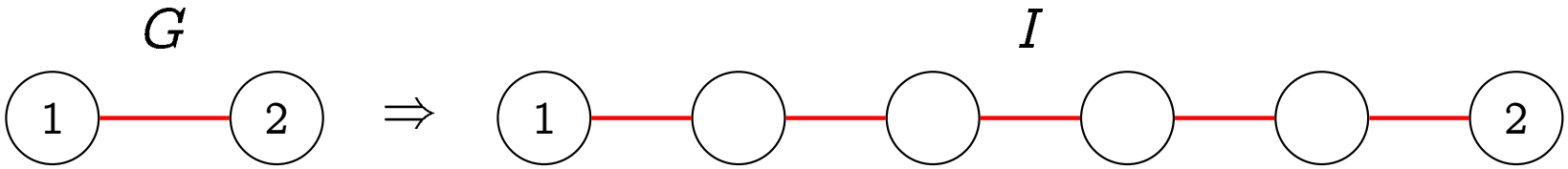
Intuition for our result: reduction

- Choose a problem from which to reduce:
  - Must be #P-hard on planar degree-3 graphs
  - Must be encodable to an FO query q (more later)
  - We use the problem of counting matchings
- Given an input graph G, compute $k := |G|^c$
- Compute in PTIME an instance I of $\mathcal{I}$ of treewidth $\geq k$
- Compute in randomized PTIME an embedding of G in I

Intuition for our result: reduction

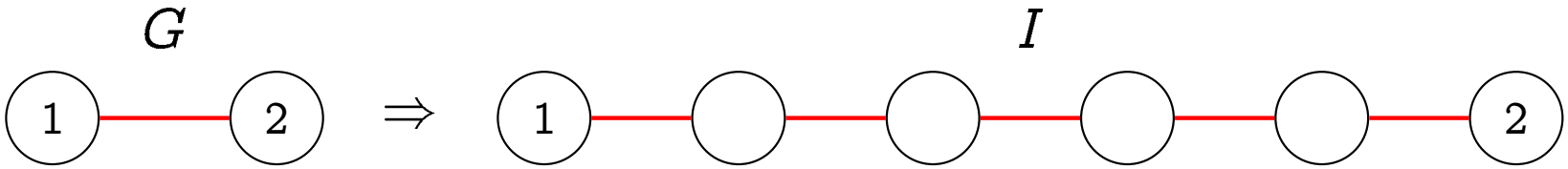
- Choose a problem from which to reduce:
  - Must be #P-hard on planar degree-3 graphs
  - Must be encodable to an FO query q (more later)
  - We use the problem of counting matchings
- Given an input graph G, compute $k := |G|^c$
- Compute in PTIME an instance I of $\mathcal{I}$ of treewidth $\geq k$
- Compute in randomized PTIME an embedding of G in I
- Construct a probability valuation π of I such that:
  - Unneccessary edges of I are removed
  - PQE for q gives the answer to the hard problem

Technical issue



- In the embedding, edges of G can become long paths in I
- q must answer the hard problem on G despite subdivisions

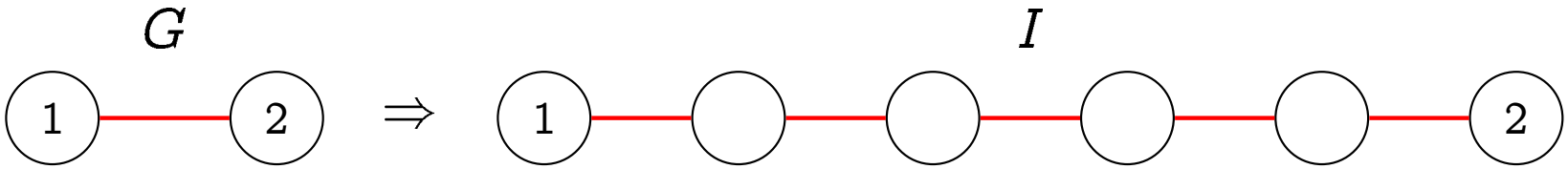
Technical issue



- In the embedding, edges of G can become long paths in I
- q must answer the hard problem on G despite subdivisions

- Our q restricts to a subset of the worlds of known weight and gives the right answer up to renormalization

Technical issue



- In the embedding, edges of G can become long paths in I
- q must answer the hard problem on G despite subdivisions

- Our q restricts to a subset of the worlds of known weight and gives the right answer up to renormalization

- For non-probabilistic evaluation, using FO does not work (Frick and Grohe, 2001)
- Lower bounds for non-probabilistic evaluation are for MSO (Ganian et al., 2014)

Can we do better?

- We can use a non-monotone FO or a monotone MSO query
- Can we use a weaker query language? (e.g., monotone FO)

Can we do better?

- We can use a non-monotone FO or a monotone MSO query
- Can we use a weaker query language? (e.g., monotone FO)

- We cannot use a connected CQ even with inequalities
- We cannot use a query closed under homomorphisms

Can we do better?

- We can use a non-monotone FO or a monotone MSO query
- Can we use a weaker query language? (e.g., monotone FO)

- We cannot use a connected CQ even with inequalities
- We cannot use a query closed under homomorphisms

- A good candidate query:

$$q : (E(x, y) \vee E(y, x)) \wedge (E(y, z) \wedge E(z, y)) \wedge x \neq z$$

Can we do better?

- We can use a non-monotone FO or a monotone MSO query
- Can we use a weaker query language? (e.g., monotone FO)

- We cannot use a connected CQ even with inequalities
- We cannot use a query closed under homomorphisms

- A good candidate query:

$$q : (E(x, y) \vee E(y, x)) \wedge (E(y, z) \wedge E(z, y)) \wedge x \neq z$$

- This UCQ with inequalities is hard in a weaker sense (no polynomial-size OBDD representations of provenance)
- We don't know whether it's #P-hard (because of subdivisions)

Outline

Practical implications

- Exploiting low treewidth is the only way to have efficient PQE beyond safe queries

Practical implications

- Exploiting low treewidth is the only way to have efficient PQE beyond safe queries
- Are real-world databases low-treewidth?

Practical implications

- Exploiting low treewidth is the only way to have efficient PQE beyond safe queries
- Are real-world databases low-treewidth?
- If not, can we still do something with them?

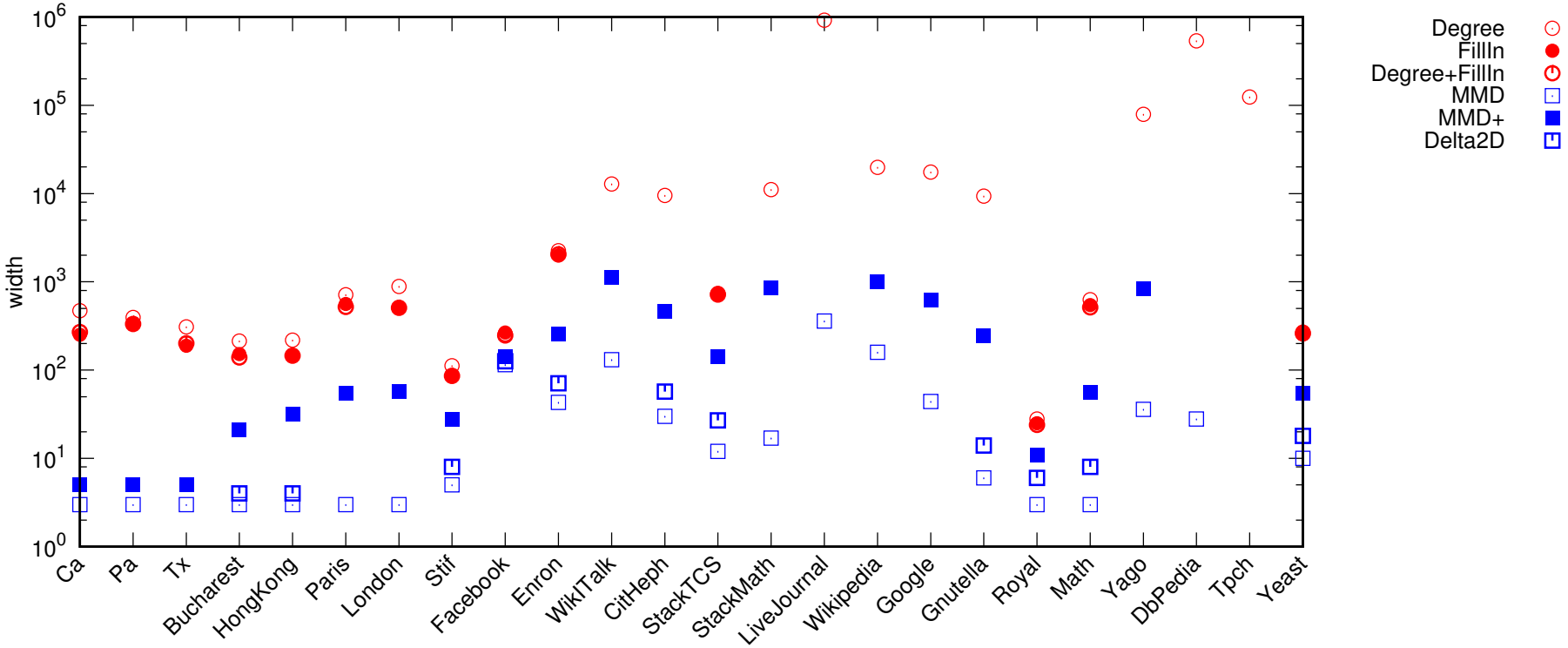
Practical implications

- Exploiting low treewidth is the only way to have efficient PQE beyond safe queries
- Are real-world databases low-treewidth?
- If not, can we still do something with them?
- What about approximation algorithms?

Computing the treewidth

- Even computing the treewidth is hard (Arnborg et al., 1987)
- But we can find upper bounds (Bodlaender and Koster, 2010) and lower bounds (Bodlaender and Koster, 2011) on treewidth relatively efficiently
- When we have a bound on the treewidth, we can find a tree decomposition in linear-time (Bodlaender, 1996)...
- but this algorithm is too costly in practice. Better using upper bound algorithms that also provide a tree decomposition

Real-world databases and treewidth (Maniu et al., 2017b)



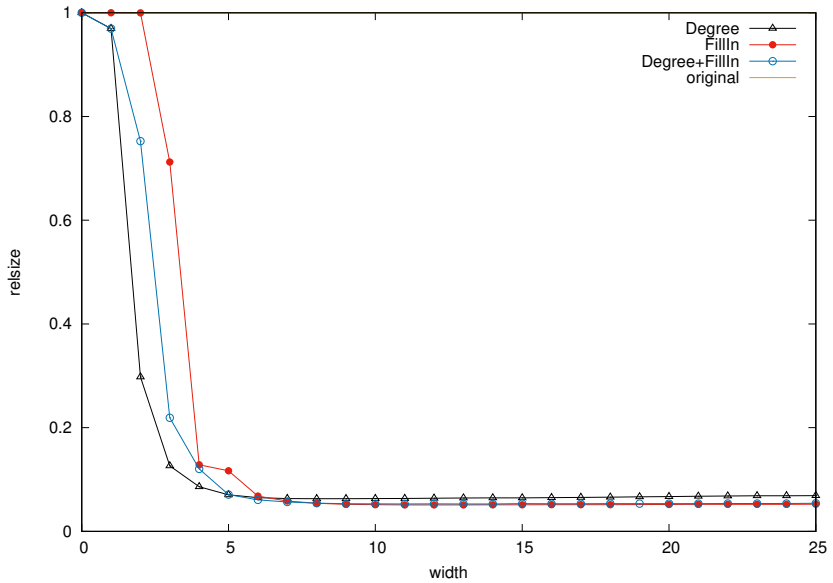
Partial tree decompositions (Maniu et al., 2017a)

If a database has high-treewidth, possible to:

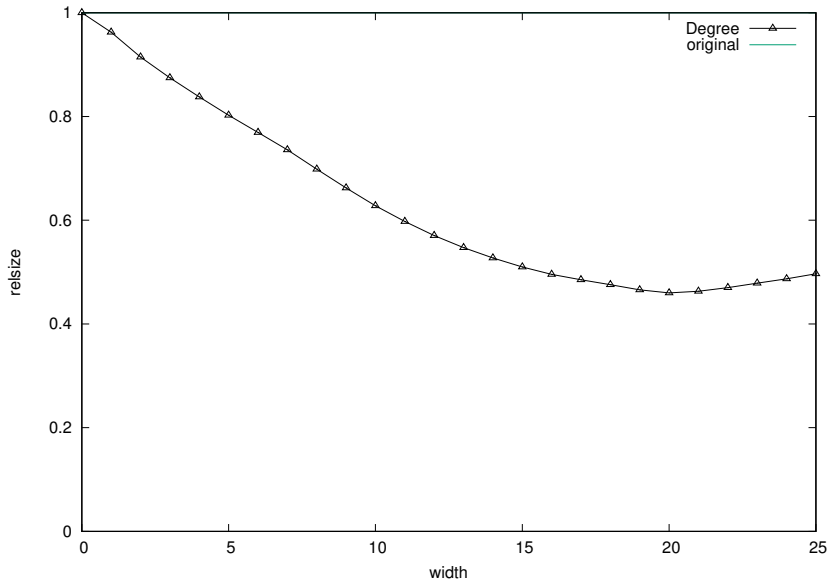
- Isolate a part of low treewidth
- Process this part with efficient PQE techniques
- Process the high-treewidth part (+ whatever is needed to keep track of the low-treewidth part) with other techniques (e.g., approximation algorithms)
- Combine results in a well-founded manner

Partial tree-decomposition of real-world databases

(Maniu et al., 2017b)



OpenStreetMaps Paris
(5m road segments)



Google Web graph fragment
(4m hyperlinks)

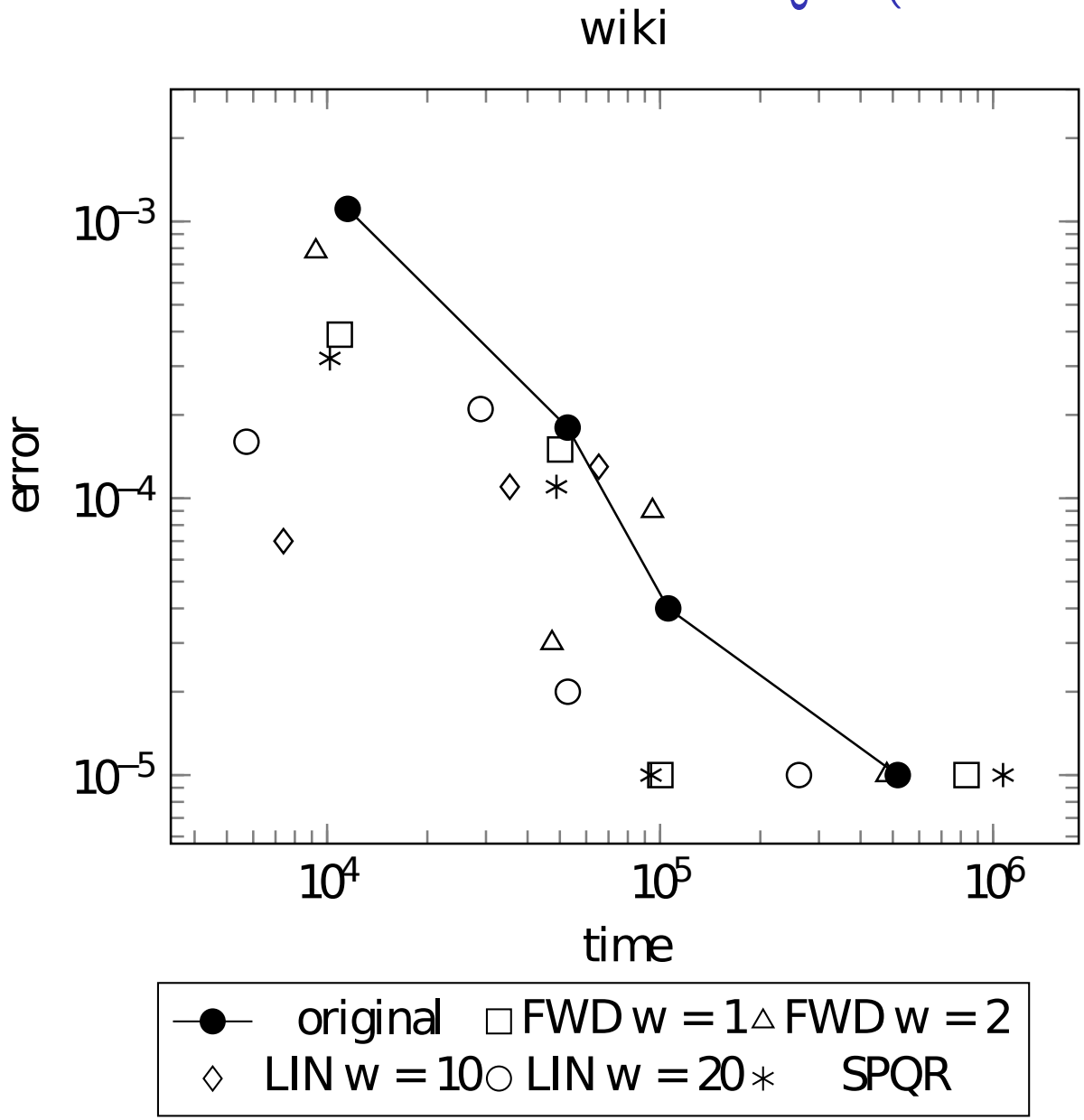
Example of query: connectedness (Maniu et al., 2017a)

- Partial tree decomposition with:
  - tendrils of low-treewidth
  - a root node of high-treewidth

Example of query: connectedness (Maniu et al., 2017a)

- Partial tree decomposition with:
  - tendrils of low-treewidth
  - a root node of high-treewidth
- Algorithm for PQE for the connectedness query:
  - Process the tree decomposition bottom-up, keeping track of the provenance of connectedness between exported nodes
  - Add virtual edges with this provenance as annotation
  - When one reaches the core, use Monte-Carlo sampling to approximate the probability

Performance of connectedness PQE (Maniu et al., 2017a)



Approximation algorithms

- Always possible to use Monte-Carlo sampling; provide an additive approximation guarantee

Approximation algorithms

- Always possible to use Monte-Carlo sampling; provide an additive approximation guarantee
- Sometimes (but not always!), possible to use more elaborate approximation algorithms, giving multiplicative approximations (Karp et al., 1989)

Approximation algorithms

- Always possible to use Monte-Carlo sampling; provide an additive approximation guarantee
- Sometimes (but not always!), possible to use more elaborate approximation algorithms, giving multiplicative approximations (Karp et al., 1989)
- Practical strategy: use query optimization strategies to decide which approximation method to use, possibly on different parts of the data/provenance (Souihli and Senellart, 2013)

Outline

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances
has linear data complexity up to arithmetic costs

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances
has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances

has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances

has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances
has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Lower. PQE for FO on any tw-constructible, arity-2, unbounded-tw instance family is #P-hard under RP reductions

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Lower. PQE for FO on any tw-constructible, arity-2, unbounded-tw instance family is #P-hard under RP reductions

Practical. In practice:

- Real-world databases rarely have low treewidth

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Lower. PQE for FO on any tw-constructible, arity-2, unbounded-tw instance family is #P-hard under RP reductions

Practical. In practice:

- Real-world databases rarely have low treewidth
- More commonly, a low-treewidth part can be isolated

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Lower. PQE for FO on any tw-constructible, arity-2, unbounded-tw instance family is #P-hard under RP reductions

Practical. In practice:

- Real-world databases rarely have low treewidth
- More commonly, a low-treewidth part can be isolated
- Possible to exploit partial tree decompositions for PQE

Summary (and generalization) of our results

Upper. PQE for MSO on treelike instances has linear data complexity up to arithmetic costs

- Also for bounded-treewidth correlations
- Can compute a provenance circuit in linear time
  - Also $\mathbb{N}[X]$-provenance (Green et al., 2007) circuits for UCQ

Lower. PQE for FO on any tw-constructible, arity-2, unbounded-tw instance family is #P-hard under RP reductions

Practical. In practice:

- Real-world databases rarely have low treewidth
- More commonly, a low-treewidth part can be isolated
- Possible to exploit partial tree decompositions for PQE
- One can always resort to approximation algorithms

Open questions and future work

- Query-specific tree decomposition or instances

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?
- Can we show #P-hardness under usual P reductions?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?
- Can we show #P-hardness under usual P reductions?
- Can we prove the lower-bound for arbitrary arity?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?
- Can we show #P-hardness under usual P reductions?
- Can we prove the lower-bound for arbitrary arity?
- Is there a monotone FO query where probability evaluation is hard on any constructible unbounded-treewidth family?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?
- Can we show #P-hardness under usual P reductions?
- Can we prove the lower-bound for arbitrary arity?
- Is there a monotone FO query where probability evaluation is hard on any constructible unbounded-treewidth family?
- Can we extend the algorithm on partial tree decomposition to more interesting query languages (regular path queries)? To more general notions of provenance?

Open questions and future work

- Query-specific tree decomposition or instances
- Better understand the connection to query-based dichotomy?
- Better understand the connection to Monadic Datalog approaches (Gottlob et al., 2010) for linear combined QE?
- Can we show #P-hardness under usual P reductions?
- Can we prove the lower-bound for arbitrary arity?
- Is there a monotone FO query where probability evaluation is hard on any constructible unbounded-treewidth family?
- Can we extend the algorithm on partial tree decomposition to more interesting query languages (regular path queries)? To more general notions of provenance?
- Can we apply all of this to a real-world problem? Routing in public transport networks with a model of uncertainty on schedules?

References I

Amarilli, A., Bourhis, P., Monet, M., and Senellart, P. (2017a). Combined tractability of query evaluation via tree automata and cycluits. In *Proc. ICDT*, pages 6:1–6:19, Venice, Italy.

Amarilli, A., Bourhis, P., and Senellart, P. (2015). Provenance circuits for trees and treelike instances. In *Proc. ICALP*, pages 56–68, Kyoto, Japan.

Amarilli, A., Bourhis, P., and Senellart, P. (2016). Tractable lineages on treelike instances: Limits and extensions. In *Proc. PODS*, pages 355–370, San Francisco, USA.

Amarilli, A., Monet, M., and Senellart, P. (2017b). Conjunctive queries on probabilistic graphs: Combined complexity. In *Proc. PODS*, San Francisco, USA.

References II

Arnborg, S., Corneil, D. G., and Proskuworski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284.

Bodlaender, H. L. (1996). A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317.

Bodlaender, H. L. and Koster, A. M. C. A. (2010). Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):345–353.

Bodlaender, H. L. and Koster, A. M. C. A. (2011). Treewidth computations II. Lower bounds. *Information and Computation*, 209(7):1103–1119.

Chaudhuri, S. and Vardi, M. Y. (1992). On the equivalence of recursive and nonrecursive Datalog programs. In *PODS*.

References III

Chekuri, C. and Chuzhoy, J. (2014). Polynomial bounds for the grid-minor theorem. In *STOC*.

Cohen, S., Kimelfeld, B., and Sagiv, Y. (2009). Running tree automata on probabilistic XML. In *PODS*.

Courcelle, B. (1990). The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1).

Dalvi, N. and Suciu, D. (2012). The dichotomy of probabilistic inference for unions of conjunctive queries. *JACM*, 59(6):30.

Darwiche, A. (2001). On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Applied Non-Classical Logics*, 11(1-2).

Deutch, D., Milo, T., Roy, S., and Tannen, V. (2014). Circuits for datalog provenance. In *ICDT*.

References IV

Frick, M. and Grohe, M. (2001). Deciding first-order properties of locally tree-decomposable structures. *JACM*, 48(6).

Ganian, R., Hliněný, P., Langer, A., Obdržálek, J., Rossmanith, P., and Sikdar, S. (2014). Lower bounds on the complexity of MSO1 model-checking. *JCSS*, 1(80).

Gottlob, G., Pichler, R., and Wei, F. (2010). Monadic datalog over finite structures of bounded treewidth. *TOCL*, 12(1):3.

Green, T. J., Karvounarakis, G., and Tannen, V. (2007). Provenance semirings. In *PODS*.

Karp, R. M., Luby, M., and Madras, N. (1989). Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3).

References V

Lauritzen, S. L. and Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society. Series B.*

Makowsky, J. A. and Marino, J. (2003). Tree-width and the monadic quantifier hierarchy. *Theor. Comput. Sci.*, 303(1).

Maniu, S., Cheng, R., and Senellart, P. (2017a). An indexing framework for queries on probabilistic graphs. *ACM Transactions on Database Systems*, 42(2).

Maniu, S., Senellart, P., and Jog, S. (2017b). Under review.

Riedel, M. D. and Bruck, J. (2012). Cyclic Boolean circuits. *Discrete Applied Mathematics*, 160(13-14).

References VI

Robertson, N. and Seymour, P. D. (1986). Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1).

Souihli, A. and Senellart, P. (2013). Optimizing approximations of dnf query lineage in probabilistic xml. In *ICDE*.

Thatcher, J. W. and Wright, J. B. (1968). Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2(1):57–81.

Encoding treelike instances (Chaudhuri and Vardi, 1992)

Instance:

N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

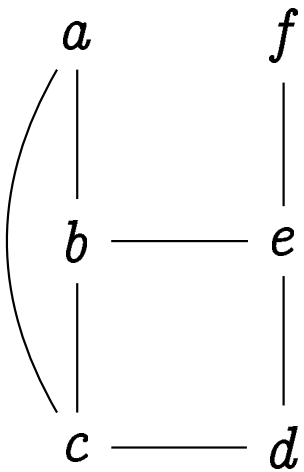
Encoding treelike instances (Chaudhuri and Vardi, 1992)

Instance:

N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

Gaifman graph:



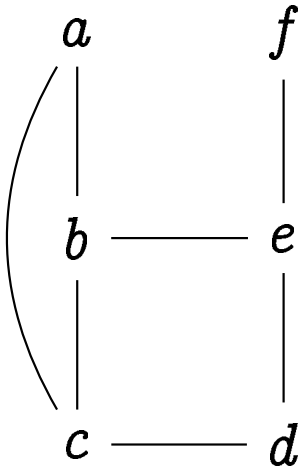
Encoding treelike instances (Chaudhuri and Vardi, 1992)

Instance:

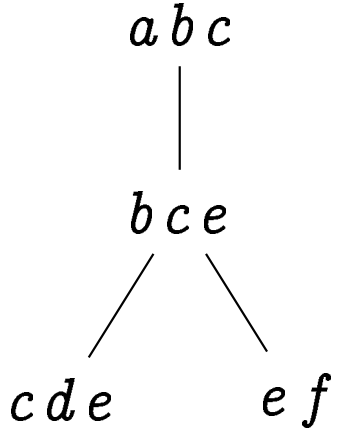
N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

Gaifman graph:



Tree decomp.:



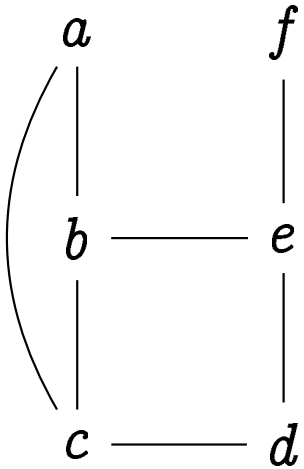
Encoding treelike instances (Chaudhuri and Vardi, 1992)

Instance:

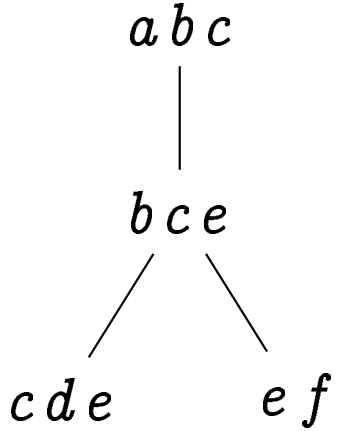
N	
a	b
b	c
c	d
d	e
e	f

S	
a	c
b	e

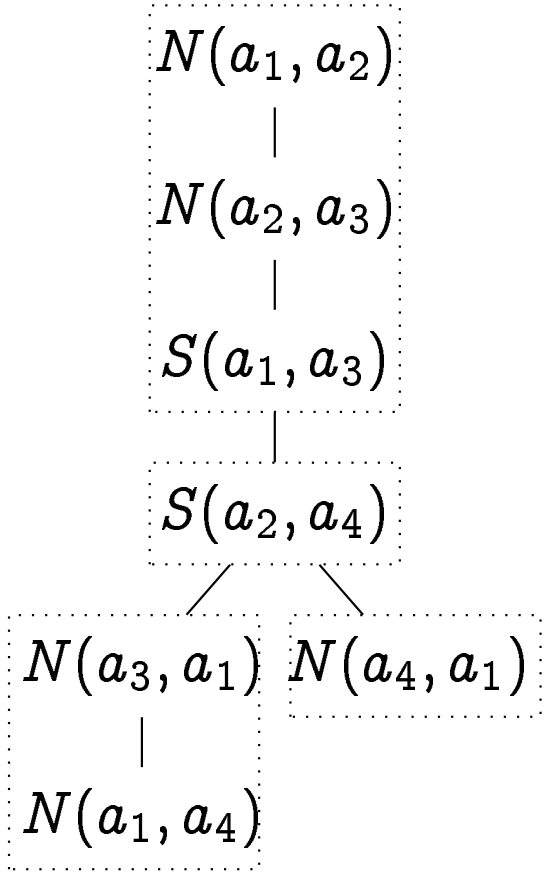
Gaifman graph:



Tree decomp.:



Tree encoding:



Outline

Provenance semirings

- Semiring of positive Boolean functions $(\mathrm{PosBool}[X], \vee, \wedge, \mathfrak{f}, \mathfrak{t})$

Provenance semirings

- Semiring of positive Boolean functions $(\mathrm{PosBool}[X], \vee, \wedge, \mathfrak{f}, \mathfrak{t})$

- Provenance semirings: (Green et al., 2007)
  - Provenance generalized to arbitrary (commutative) semirings
  - For queries in the positive relational algebra and Datalog

Provenance semirings

- Semiring of positive Boolean functions $(\mathrm{PosBool}[X], \vee, \wedge, \mathfrak{f}, \mathfrak{t})$
- Provenance semirings: (Green et al., 2007)
  - Provenance generalized to arbitrary (commutative) semirings
  - For queries in the positive relational algebra and Datalog
- Our circuits capture $\mathrm{PosBool}[X]$-provenance in this sense

Provenance semirings

- Semiring of positive Boolean functions $(\mathrm{PosBool}[X], \vee, \wedge, \mathfrak{f}, \mathfrak{t})$
- Provenance semirings: (Green et al., 2007)
  - Provenance generalized to arbitrary (commutative) semirings
  - For queries in the positive relational algebra and Datalog
- Our circuits capture $\mathrm{PosBool}[X]$-provenance in this sense
  - The definitions match: all subinstances that satisfy the query

Provenance semirings

- Semiring of positive Boolean functions $(\mathrm{PosBool}[X], \vee, \wedge, \mathfrak{f}, \mathfrak{t})$
- Provenance semirings: (Green et al., 2007)
  - Provenance generalized to arbitrary (commutative) semirings
  - For queries in the positive relational algebra and Datalog
- Our circuits capture $\mathrm{PosBool}[X]$-provenance in this sense
  - The definitions match: all subinstances that satisfy the query
  - For monotone queries, we can construct positive circuits

Universal provenance

- Universal semiring of polynomials $(\mathbb{N}[X], +, \times, 0, 1)$
  - The provenance for $\mathbb{N}[X]$ can be specialized to any $K[X]$

Universal provenance

- Universal semiring of polynomials $(\mathbb{N}[X], +, \times, 0, 1)$
  - The provenance for $\mathbb{N}[X]$ can be specialized to any $K[X]$
- Captures many useful semirings:
  - counting the number of matches of a query
  - computing the security level of a query result
  - computing the cost of a query result

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:
- $\mathbb{N}[X]$-provenance:

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x, y) \wedge R(y, z)$$

- PosBool$[X]$-provenance:
- $\mathbb{N}[X]$-provenance:

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \qquad \vee \quad x_5$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3) + (x_5 \times x_5)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ \ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \qquad \vee \quad x_5$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3) + (x_5 \times x_5)$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ \ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \qquad \vee \quad x_5$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3) + (x_5 \times x_5)$$
$$= x_1x_2 + 2x_3x_4 + x_5^2$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

$\mathbb{N}[X]$-provenance example

R		
a	b	x_1
b	c	x_2
d	e	x_3
e	d	x_4
f	f	x_5

$$\exists x\, y\, z\ R(x,y) \wedge R(y,z)$$

- PosBool$[X]$-provenance:

$$(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \quad \vee \quad x_5$$

- $\mathbb{N}[X]$-provenance:

$$(x_1 \times x_2) + (x_3 \times x_4) + (x_4 \times x_3) + (x_5 \times x_5)$$
$$= x_1x_2 + 2x_3x_4 + x_5^2$$

- Definition of provenance for conjunctive queries:
  - Sum over query matches
  - Multiply over matched facts

How is $\mathbb{N}[X]$ more expressive than PosBool$[X]$?

- Coefficients: counting multiple matches
- Exponents: using facts multiple times

Capturing $\mathbb{N}[X]$-provenance

Our construction can be extended to $\mathbb{N}[X]$-provenance for conjunctive queries and unions of conjunctive queries (UCQ):

Capturing $\mathbb{N}[X]$-provenance

Our construction can be extended to $\mathbb{N}[X]$-provenance for conjunctive queries and unions of conjunctive queries (UCQ):

Theorem

For any fixed UCQ q and $k \in \mathbb{N}$, for any input instance I of treewidth $\leq k$, we can build in linear time a $\mathbb{N}[X]$-provenance circuit of q on I.

Capturing $\mathbb{N}[X]$-provenance

Our construction can be extended to $\mathbb{N}[X]$-provenance for conjunctive queries and unions of conjunctive queries (UCQ):

Theorem

For any fixed UCQ q and $k \in \mathbb{N}$, for any input instance I of treewidth $\leq k$, we can build in linear time a $\mathbb{N}[X]$-provenance circuit of q on I.

- What fails for MSO and Datalog?
  - Unbounded maximal multiplicity of fact uses

Outline

Correlations

- Our probabilistic instances assume independence on all facts
  - Not very expressive!

Correlations

- Our probabilistic instances assume independence on all facts
  - Not very expressive!

More expressive formalism: Block-Independent Disjoint instances:

name	**city**	**iso**	p
pods	san francisco	us	0.8
pods	los angeles	us	0.2
icalp	rome	it	0.1
icalp	florence	it	0.9

pc-tables

More generally, pc-tables to represent arbitrary correlations

pc-tables

More generally, pc-tables to represent arbitrary correlations

date	teacher	room	
04	John	C42	$\neg x_1$
04	Jane	C42	x_1
11	John	C017	$x_2 \wedge \neg x_1$
11	Jane	C017	$x_2 \wedge x_1$
11	John	C47	$\neg x_2 \wedge \neg x_1$
11	Jane	C47	$\neg x_2 \wedge x_1$

pc-tables

More generally, pc-tables to represent arbitrary correlations

date	teacher	room	
04	John	C42	$\neg x_1$
04	Jane	C42	x_1
11	John	C017	$x_2 \wedge \neg x_1$
11	Jane	C017	$x_2 \wedge x_1$
11	John	C47	$\neg x_2 \wedge \neg x_1$
11	Jane	C47	$\neg x_2 \wedge x_1$

x_1 John gets sick

- Probability 0.1

x_2 Room C017 is available

- Probability 0.2

Our results

Probabilistic query evaluation on instances with correlations is tractable if the instance and correlations are bounded-tw:

Our results

Probabilistic query evaluation on instances with correlations is tractable if the instance and correlations are bounded-tw:

Theorem

Probabilistic query evaluation of MSO queries on treelike BID is in linear time up to arithmetic operations.

Our results

Probabilistic query evaluation on instances with correlations is tractable if the instance and correlations are bounded-tw:

Theorem

Probabilistic query evaluation of MSO queries on treelike BID

is in linear time up to arithmetic operations.

"Tree-like" just means the underlying instance (easy correlations)

Our results

Probabilistic query evaluation on instances with correlations is tractable if the instance and correlations are bounded-tw:

Theorem

Probabilistic query evaluation of MSO queries on treelike BID is in linear time up to arithmetic operations.

"Tree-like" just means the underlying instance (easy correlations)

Theorem

Probabilistic query evaluation of MSO queries on treelike pc-tables is in linear time up to arithmetic operations.

Our results

Probabilistic query evaluation on instances with correlations is tractable if the instance and correlations are bounded-tw:

Theorem

Probabilistic query evaluation of MSO queries on treelike BID is in linear time up to arithmetic operations.

"Tree-like" just means the underlying instance (easy correlations)

Theorem

Probabilistic query evaluation of MSO queries on treelike pc-tables is in linear time up to arithmetic operations.

"Tree-like" refers to the underlying instance, adding facts to represent variable occurrences and co-occurrences