Knowledge Compilation for Probabilistic Databases

Mikaël Monet[1,2]

November 3rd, 2017

[1]LTCI, Télécom ParisTech, Université Paris-Saclay; Paris, France

[2]Inria Paris; Paris, France

Tuple-independent databases (TID)

- Probabilistic databases: model uncertainty about data
- Simplest model: tuple-independent databases (TID)
  - A relational database I
  - A probability valuation π mapping each fact of I to $[0, 1]$
- Semantics of a TID (I, π): a probability distribution on $I' \subseteq I$:
  - Each fact $F \in I$ is either present or absent with probability $\pi(F)$
  - Assume independence across facts

Example: TID

	S	
a	a	.5
b	c	.2

Example: TID

S		
a	a	.5
b	c	.2

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

Example: TID

S		
a	a	.5
b	c	.2

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

$.5 \times .2$

S	
a	a
b	c

Example: TID

S		
a	a	.5
b	c	.2

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

$.5 \times .2$	
S	
a	a
b	c

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

Example: TID

S		
a	a	.5
b	c	.2

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

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

Example: TID

S		
a	a	.5
b	c	.2

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

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

Probabilistic query evaluation (PQE)

Let us fix:

- Relational signature σ
- A Boolean query q (e.g., CQ, FO, Datalog...)
- Class $\mathcal{I}$ of relational instances on σ (e.g., acyclic, treelike)

Probabilistic query evaluation (PQE)

Let us fix:

- Relational signature σ
- A Boolean query q (e.g., CQ, FO, Datalog...)
- Class $\mathcal{I}$ of relational instances on σ (e.g., acyclic, treelike)

Probabilistic query evaluation PQE($q, \mathcal{I}$):

- **INPUT:** an instance $I \in \mathcal{I}$ and a probability valuation π
- **OUTPUT:** the probability that (I, π) satisfies q

Probabilistic query evaluation (PQE)

Let us fix:

- Relational signature σ
- A Boolean query q (e.g., CQ, FO, Datalog...)
- Class $\mathcal{I}$ of relational instances on σ (e.g., acyclic, treelike)

Probabilistic query evaluation PQE($q, \mathcal{I}$):

- **INPUT:** an instance $I \in \mathcal{I}$ and a probability valuation π
- **OUTPUT:** the probability that (I, π) satisfies q

$\rightarrow \Pr_\pi(I \models q) = \sum_{J \subseteq I,\ J \models q} \Pr_\pi(J)$

Example: PQE

$$I = \begin{array}{|ccc|}\hline \multicolumn{3}{|c|}{\mathbf{S}} \\ \hline a & a & .5 \\ b & c & .2 \\ \hline \end{array} \qquad q = \exists x\, y\; S(x, y) \wedge x \neq y$$

$\Pr_\pi(I \models q) =$

Example: PQE

$I =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$
S
a a
b c

$\Pr_\pi(I \models q) =$

Example: PQE

$$I = \begin{array}{|ccc|}\hline \multicolumn{3}{|c|}{\mathbf{S}} \\ \hline a & a & .5 \\ b & c & .2 \\ \hline \end{array} \qquad q = \exists x\, y\; S(x, y) \wedge x \neq y$$

$$\begin{array}{|cc|}\hline .5 \times .2 & \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline a & a \\ b & c \\ \hline \multicolumn{2}{|c|}{\checkmark} \\ \hline \end{array}$$

$\Pr_\pi(I \models q) = .5 \times .2$

Example: PQE

$I =$

S		
a	a	.5
b	c	.2

$q = \exists x\, y\; S(x, y) \wedge x \neq y$

$.5 \times .2$	
S	
a	a
b	c
✔	

$.5 \times (1 - .2)$	
S	
a	a
	✖

$\Pr_\pi(I \models q) = .5 \times .2$

Example: PQE

$$I = \begin{array}{|ccc|}\hline \multicolumn{3}{|c|}{\mathbf{S}} \\ \hline a & a & .5 \\ b & c & .2 \\ \hline \end{array} \qquad q = \exists x\, y\; S(x, y) \wedge x \neq y$$

$$\begin{array}{|cc|}\hline \multicolumn{2}{|c|}{.5 \times .2} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline a & a \\ b & c \\ \hline \end{array} \qquad \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{.5 \times (1 - .2)} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline a & a \\ \\ \hline \end{array} \qquad \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{(1 - .5) \times .2} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline \\ b \quad c \\ \hline \end{array}$$

✔ ✖ ✔

$$\Pr_\pi(I \models q) = .5 \times .2$$

Example: PQE

$$I = \begin{array}{|ccc|}\hline \multicolumn{3}{|c|}{\mathbf{S}} \\ \hline a & a & .5 \\ b & c & .2 \\ \hline \end{array} \qquad q = \exists x\, y\; S(x, y) \wedge x \neq y$$

$$\begin{array}{|cc|}\hline \multicolumn{2}{|c|}{.5 \times .2} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline a & a \\ b & c \\ \hline \end{array} \qquad \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{.5 \times (1 - .2)} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline a & a \\ \\ \hline \end{array} \qquad \begin{array}{|cc|}\hline \multicolumn{2}{|c|}{(1 - .5) \times .2} \\ \hline \multicolumn{2}{|c|}{\mathbf{S}} \\ \hline \\ b \quad c \\ \hline \end{array}$$

✔ ✖ ✔

$$\Pr_\pi(I \models q) = .5 \times .2 + (1 - .5) \times .2$$

Example: PQE

$$I = \begin{array}{|ccc|}\hline \multicolumn{3}{|c|}{\mathbf{S}} \\ \hline a & a & .5 \\ b & c & .2 \\ \hline \end{array} \qquad q = \exists x\, y\; S(x, y) \wedge x \neq y$$

$.5 \times .2$	$.5 \times (1 - .2)$	$(1 - .5) \times .2$	$(1 - .5) \times (1 - .2)$
S	**S**	**S**	**S**
a a	a a		
b c		b c	
✔	✘	✔	✘

$$\Pr_\pi(I \models q) = .5 \times .2 + (1 - .5) \times .2$$

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances
  - Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances
  - Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
  - $q \in$ MSO

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances
  - Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
  - $q \in$ MSO

  $\rightarrow$ $\text{PQE}(q, \mathcal{I}_k)$ has linear time complexity [Amarilli, Bourhis, & Senellart, 2015]

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances
  - Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
  - $q \in$ MSO

  $\rightarrow$ $\text{PQE}(q, \mathcal{I}_k)$ has linear time complexity [Amarilli, Bourhis, & Senellart, 2015]

  $\rightarrow$ There is an FO query q_{hard} for which $\text{PQE}(q_{\text{hard}}, \mathcal{I})$ is **#P-hard** on any unbounded-treewidth graph family $\mathcal{I}$ (under some assumptions) [Amarilli, Bourhis, & Senellart, 2016]

Intensional Method: through *Provenance*

Definition

The *provenance* $\mathsf{Prov}(q, I)$ of query q on instance I is the Boolean function with facts of I as variables and such that for any valuation $\nu : I \to \{0, 1\}$, $\mathsf{Prov}(q, I)$ evaluates to TRUE under ν iff $\{F \in I | \nu(F) = 1\} \models q$

Intensional Method: through *Provenance*

Definition

The *provenance* $\mathsf{Prov}(q, I)$ of query q on instance I is the Boolean function with facts of I as variables and such that for any valuation $\nu : I \to \{0, 1\}$, $\mathsf{Prov}(q, I)$ evaluates to TRUE under ν iff $\{F \in I | \nu(F) = 1\} \models q$

Possible representations:

- Boolean formulas (with the tuples as variables)

Intensional Method: through *Provenance*

Definition

The *provenance* $\mathsf{Prov}(q, I)$ of query q on instance I is the Boolean function with facts of I as variables and such that for any valuation $\nu : I \to \{0, 1\}$, $\mathsf{Prov}(q, I)$ evaluates to TRUE under ν iff $\{F \in I | \nu(F) = 1\} \models q$

Possible representations:

- Boolean formulas (with the tuples as variables)
- Boolean circuits

Intensional Method: through *Provenance*

Definition

The *provenance* $\mathsf{Prov}(q, I)$ of query q on instance I is the Boolean function with facts of I as variables and such that for any valuation $\nu : I \to \{0, 1\}$, $\mathsf{Prov}(q, I)$ evaluates to TRUE under ν iff $\{F \in I | \nu(F) = 1\} \models q$

Possible representations:

- Boolean formulas (with the tuples as variables)
- Boolean circuits

Then $\mathsf{Pr}_\pi(I \models q) = \mathsf{Pr}_\pi(\mathsf{Prov}(q, I) = \top)$

Example: Provenance

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

R		
b	c	.1
c	a	.1
c	d	.05

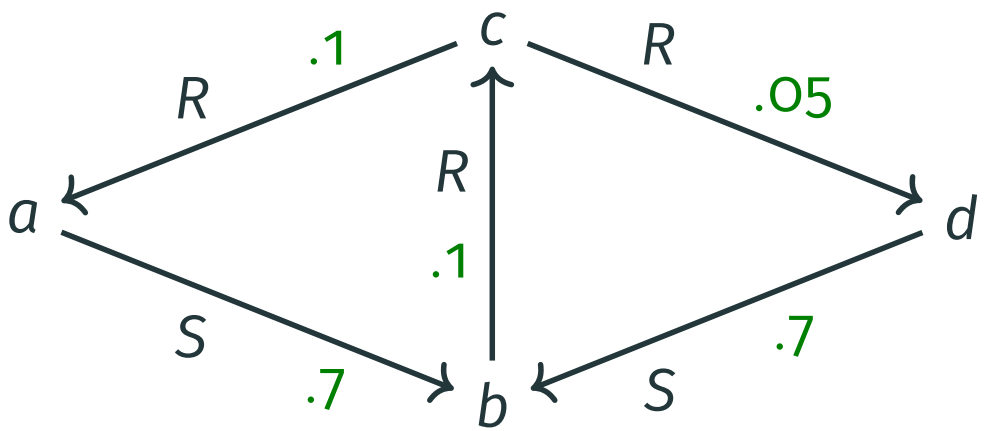
S		
a	b	.7
d	b	.7

Example: Provenance

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

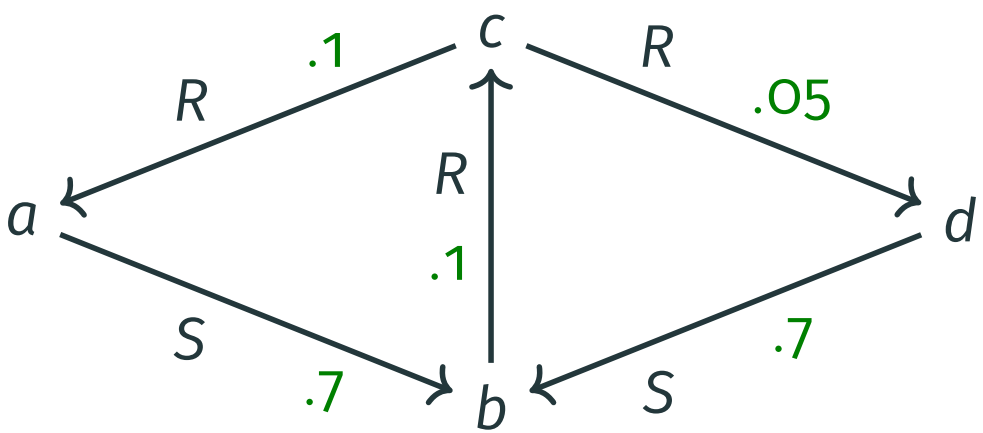
R		
b	c	.1
c	a	.1
c	d	.05

S		
a	b	.7
d	b	.7



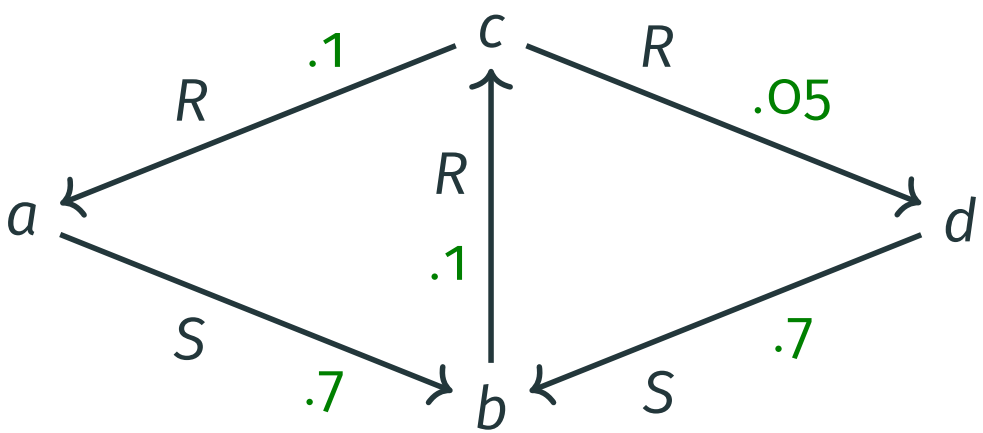
Example: Provenance

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



Example: Provenance

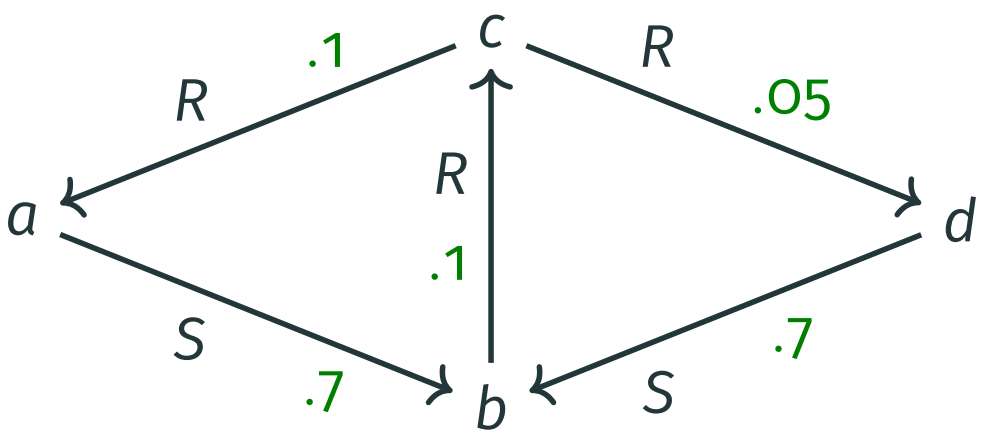
$$\exists x\, y\, z\ (R(x,y) \wedge S(y,z)) \vee (S(x,y) \wedge R(y,z))$$



$$\text{Prov}(q, I) = [S(a,b) \wedge (R(b,c) \vee R(c,a))]$$

Example: Provenance

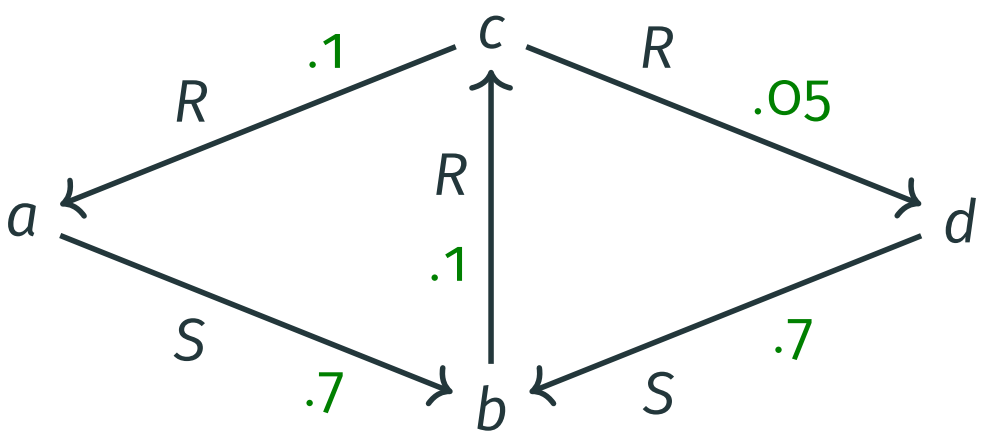
$$\exists x\, y\, z\ (R(x,y) \wedge S(y,z)) \vee (S(x,y) \wedge R(y,z))$$

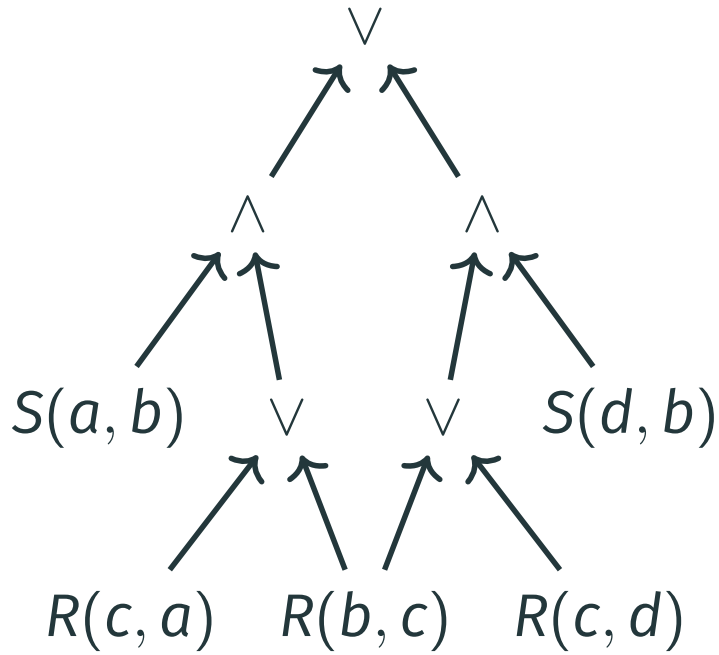


$$\begin{aligned}\mathrm{Prov}(q, I) = &[S(a,b) \wedge (R(b,c) \vee R(c,a))] \\ \vee &[S(d,b) \wedge (R(b,c) \vee R(c,d))]\end{aligned}$$

Example: Provenance

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





$$\text{Prov}(q, I) = [S(a,b) \wedge (R(b,c) \vee R(c,a))]$$
$$\vee\, [S(d,b) \wedge (R(b,c) \vee R(c,d))]$$

Ooops!

→ Computing the probability of a Boolean formula φ (or circuit C) over variables X is **#P-hard**!

Ooops!

→ Computing the probability of a Boolean formula φ (or circuit C) over variables X is **#P-hard**!

- **#SAT**: take $\pi(x) = \frac{1}{2}$ for each variable $x \in X$.
  Then $\#\varphi = 2^{|X|} \times \mathsf{Pr}_\pi(\varphi = \top)$

→ Which restrictions on φ or C make it possible?

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable...

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **evaluation**

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **enumeration**

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **SAT**

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **#SAT**

Knowledge Compilation to the Rescue

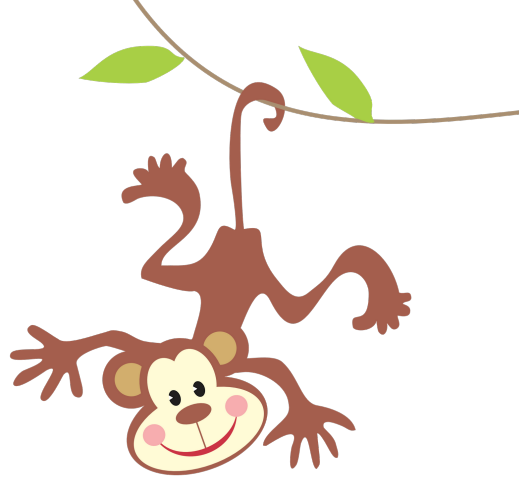
- Class of circuits $\mathcal{C} \longrightarrow$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **probability computation**

Knowledge Compilation to the Rescue

- Class of circuits $\mathcal{C} \xrightarrow{\textit{Complexity?}}$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **probability computation**
- Upper/lower complexity bounds

Knowledge Compilation to the Rescue

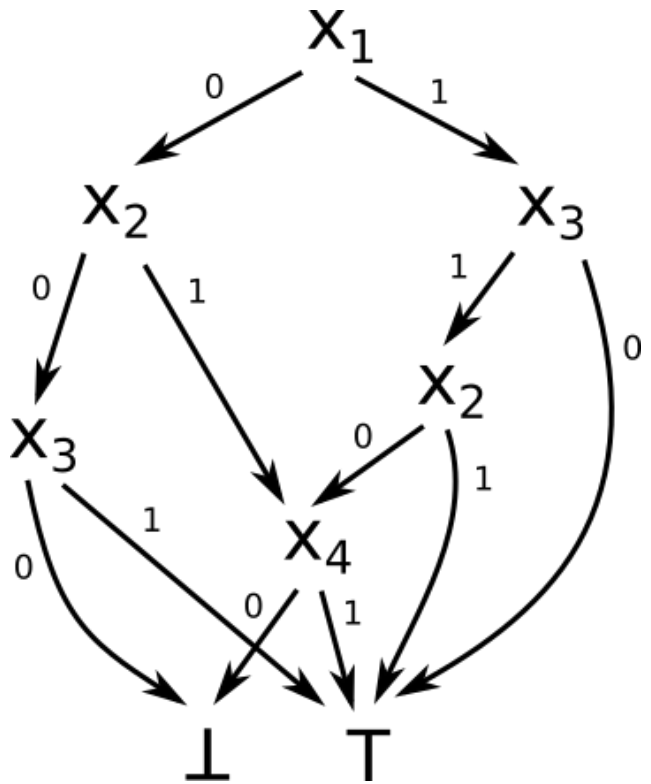
- Class of circuits $\mathcal{C} \xrightarrow{\textit{Complexity?}}$ nice representations $\mathcal{C}_{\text{target}}$
- $\mathcal{C}_{\text{target}}$ should allow tractable... **probability computation**
- Upper/lower complexity bounds
- Trade off between consiseness of $\mathcal{C}$ and $\mathcal{C}_{\text{target}}$ and complexity



Principal classes of compilation targets considered in knowledge compilation

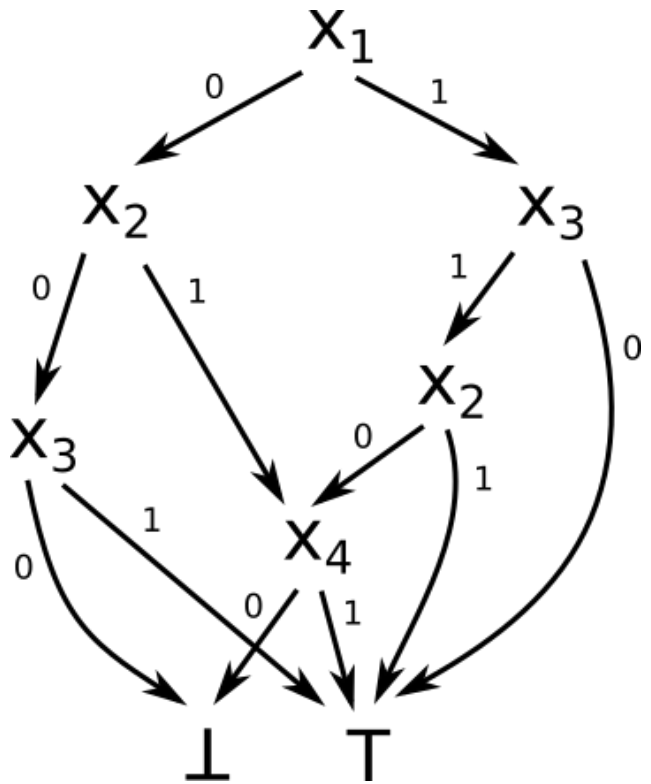
Free Binary Decision Diagrams (FBDDs)

- DAG with sink nodes $\{\top, \bot\}$ and internal nodes labeled by variables



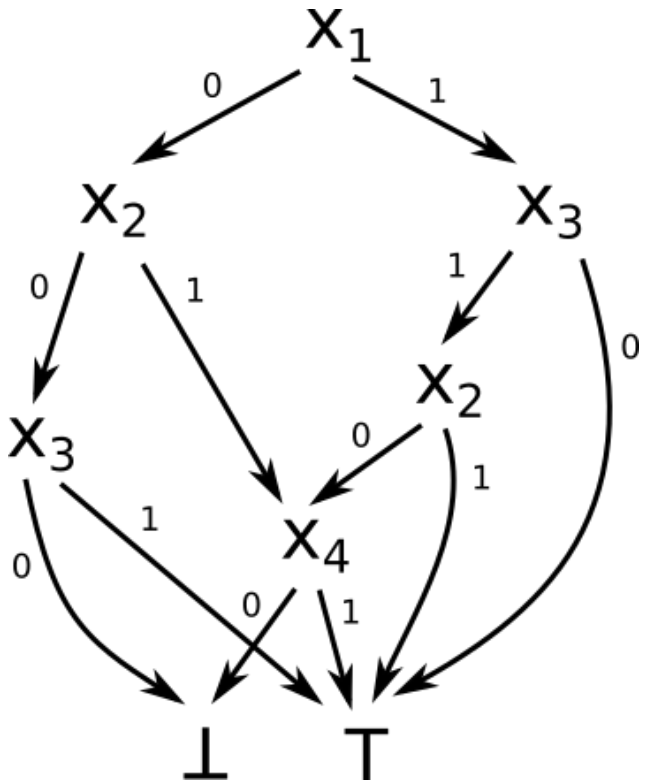
Free Binary Decision Diagrams (FBDDs)

- DAG with sink nodes $\{\top, \bot\}$ and internal nodes labeled by variables
- Each variable node has a 0 and a 1-outgoing edge



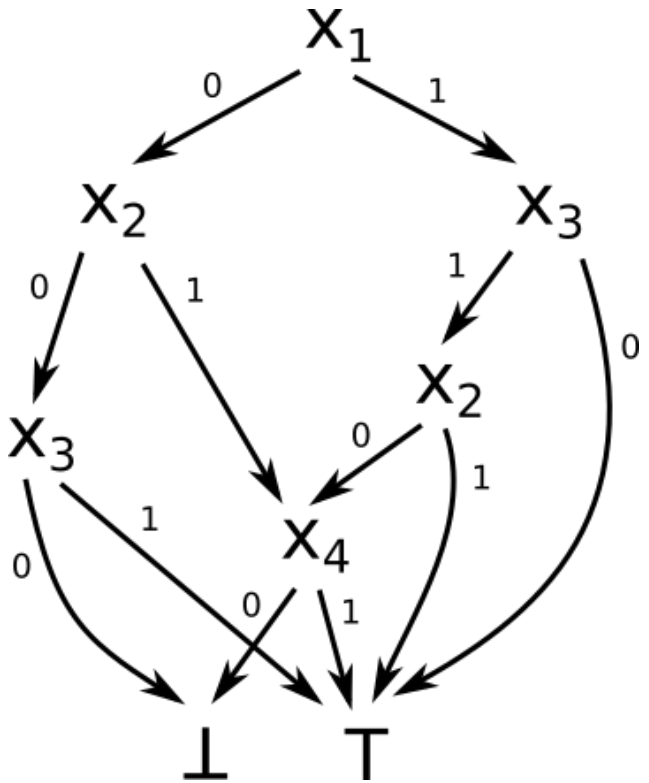
Free Binary Decision Diagrams (FBDDs)

- DAG with sink nodes $\{\top, \bot\}$ and internal nodes labeled by variables
- Each variable node has a 0 and a 1-outgoing edge
- Each root-to-sink path inspects each variable at most once



Free Binary Decision Diagrams (FBDDs)

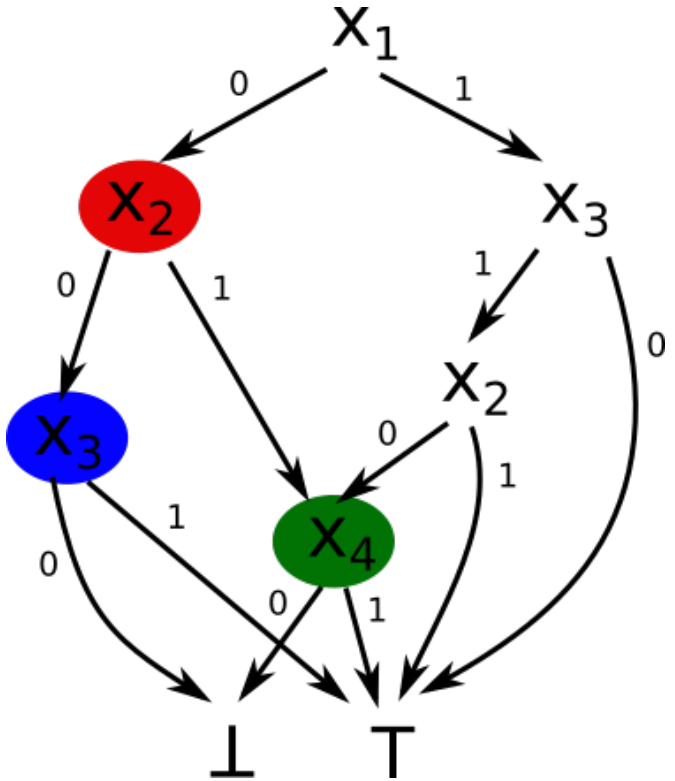
- DAG with sink nodes $\{\top, \bot\}$ and internal nodes labeled by variables
- Each variable node has a 0 and a 1-outgoing edge
- Each root-to-sink path inspects each variable at most once
- Compute probability bottom-up



Free Binary Decision Diagrams (FBDDs)

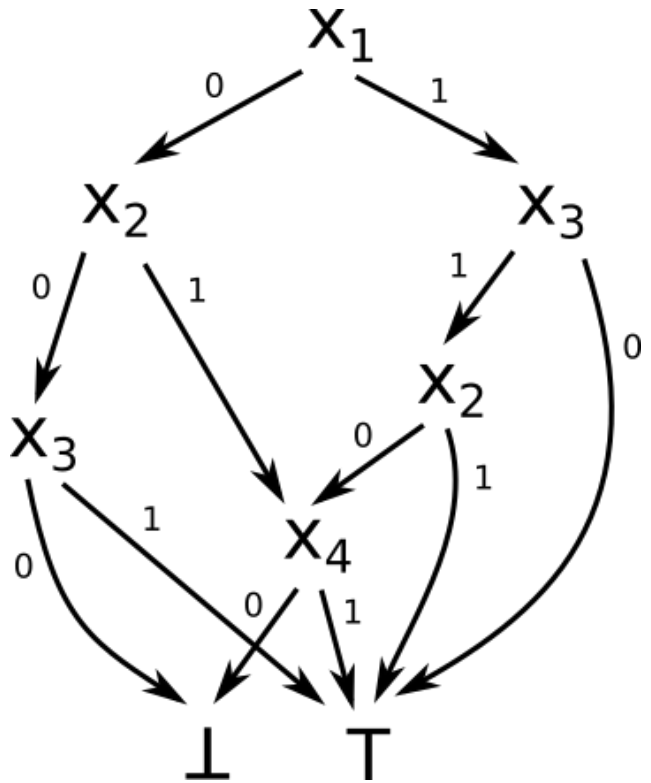
- DAG with sink nodes $\{\top, \bot\}$ and internal nodes labeled by variables
- Each variable node has a 0 and a 1-outgoing edge
- Each root-to-sink path inspects each variable at most once
- Compute probability bottom-up

$$\Pr_\pi(\bullet) = \pi(X_2) \times \Pr_\pi(\bullet) + (1 - \pi(X_2)) \times \Pr_\pi(\bullet)$$



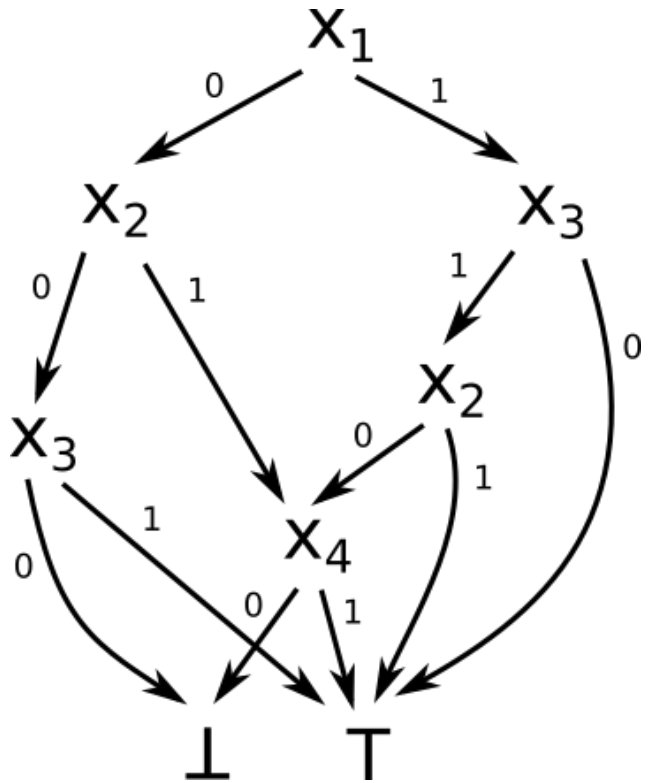
Ordered Binary Decision Diagrams (OBDDs)

- It is a FBDD



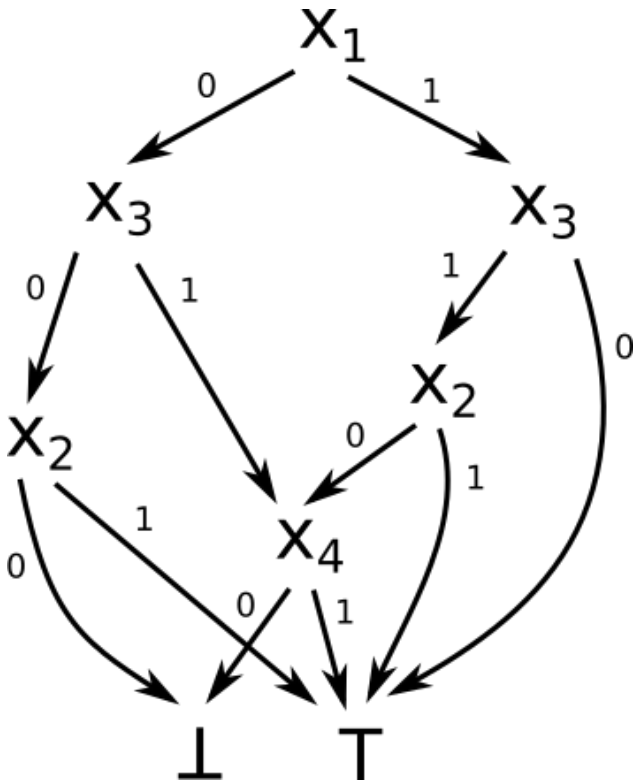
Ordered Binary Decision Diagrams (OBDDs)

- It is a FBDD
- There is a total order on the variables s.t. every root-to-sink path is compatible with this order



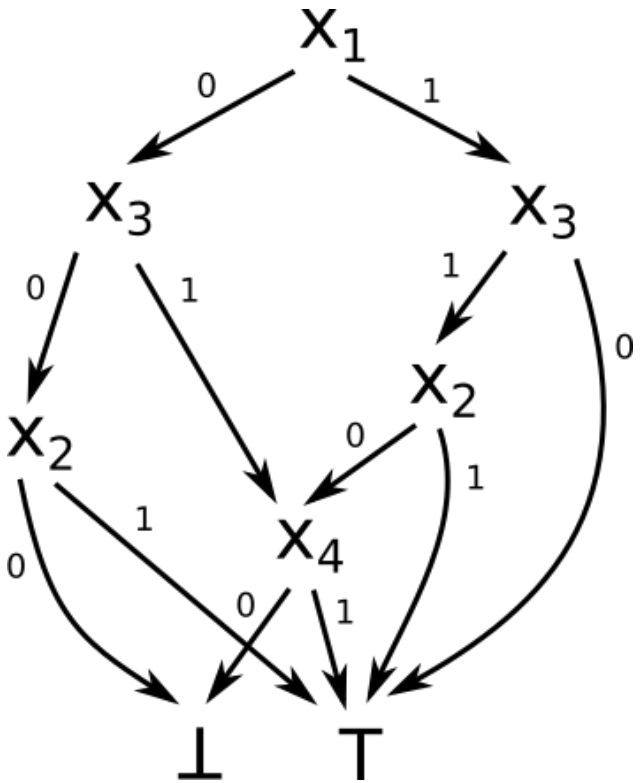
Ordered Binary Decision Diagrams (OBDDs)

- It is a FBDD
- There is a total order on the variables s.t. every root-to-sink path is compatible with this order



Ordered Binary Decision Diagrams (OBDDs)

- It is a FBDD
- There is a total order on the variables s.t. every root-to-sink path is compatible with this order
- Tractable closure under Boolean operations, assuming the orders are the same



Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

$\rightarrow$ **SAT**

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

→ **SAT**

- Deterministic DNNFs (d-DNNFs): inputs of **or**-gates are mutually exclusive

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

→ **SAT**

- Deterministic DNNFs (d-DNNFs): inputs of **or**-gates are mutually exclusive

→ **#SAT**, **probability computation**

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

→ **SAT**

- Deterministic DNNFs (d-DNNFs): inputs of **or**-gates are mutually exclusive

→ **#SAT**, **probability computation**

- Structured DNNFs (SDNNFs, d-SDNNFs): **and**-gates of the circuit are structured by a vtree

Circuit classes

- Negation Normal Forms (NNFs): negations only applied to variables
- Decomposable NNFs (DNNFs): inputs of **and**-gates are syntactically independent

→ **SAT**

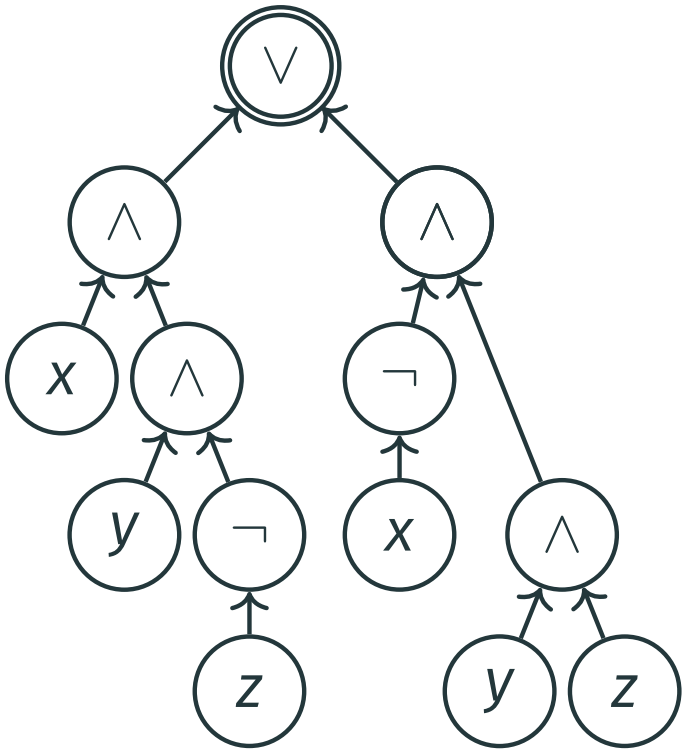
- Deterministic DNNFs (d-DNNFs): inputs of **or**-gates are mutually exclusive

→ **#SAT**, **probability computation**

- Structured DNNFs (SDNNFs, d-SDNNFs): **and**-gates of the circuit are structured by a vtree

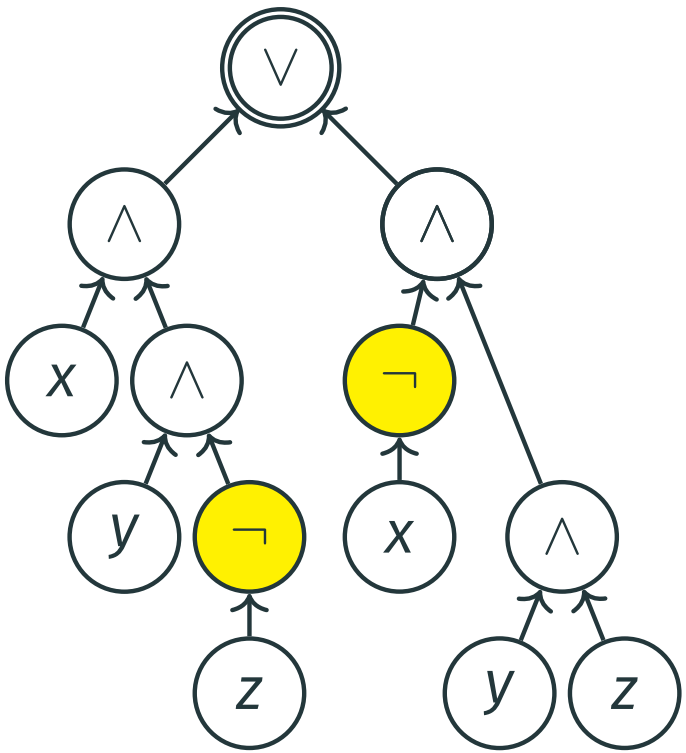
→ **Enumeration**

Example: what is this?



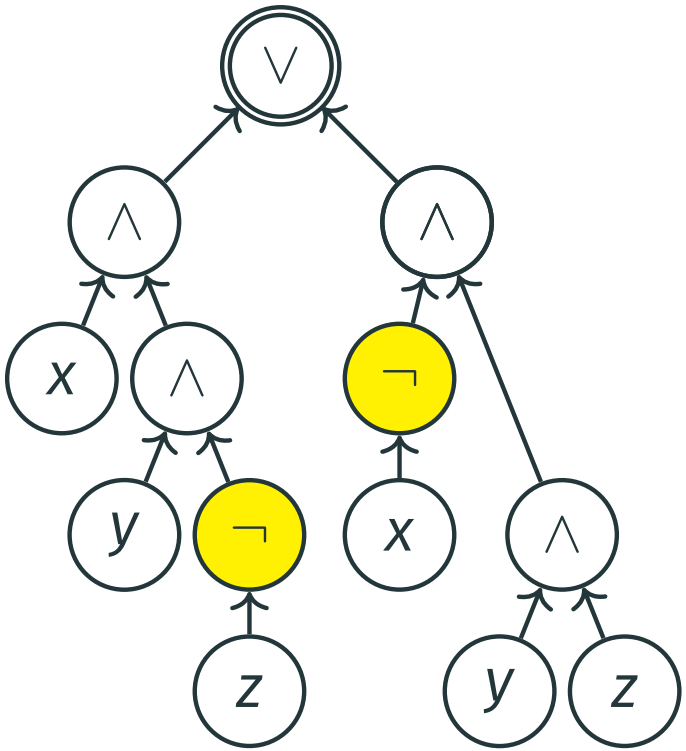
(Circuit courtesy of Antoine Amarilli)

Example: what is this?



(Circuit courtesy of Antoine Amarilli)

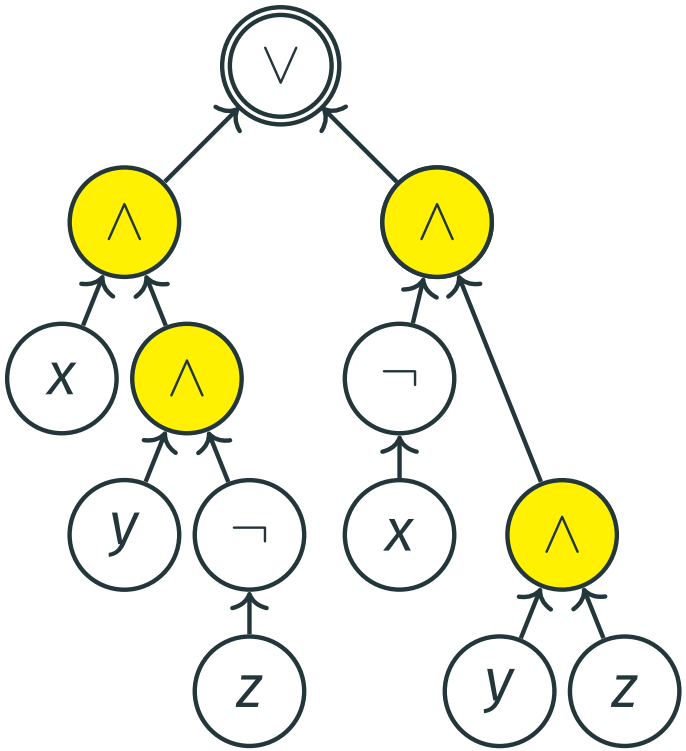
Example: what is this?



- NNF

(Circuit courtesy of Antoine Amarilli)

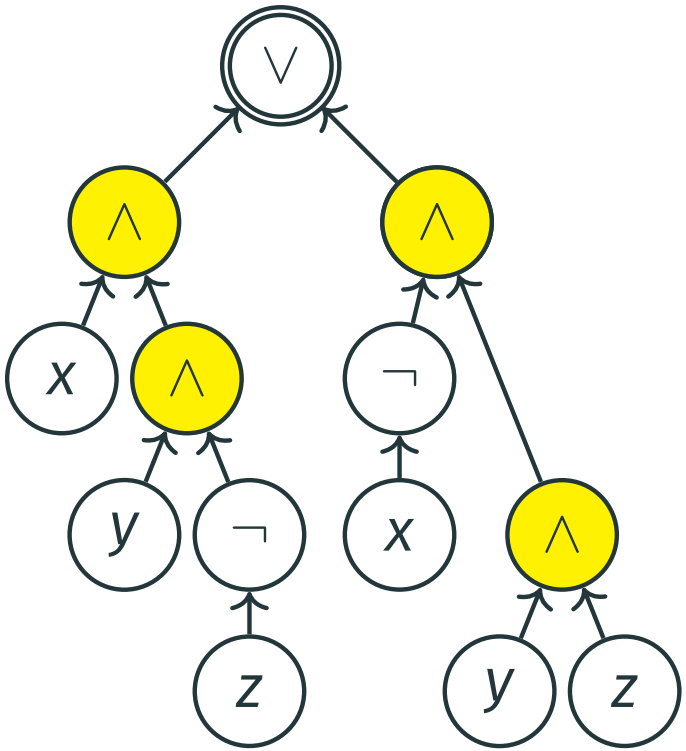
Example: what is this?



- NNF

(Circuit courtesy of Antoine Amarilli)

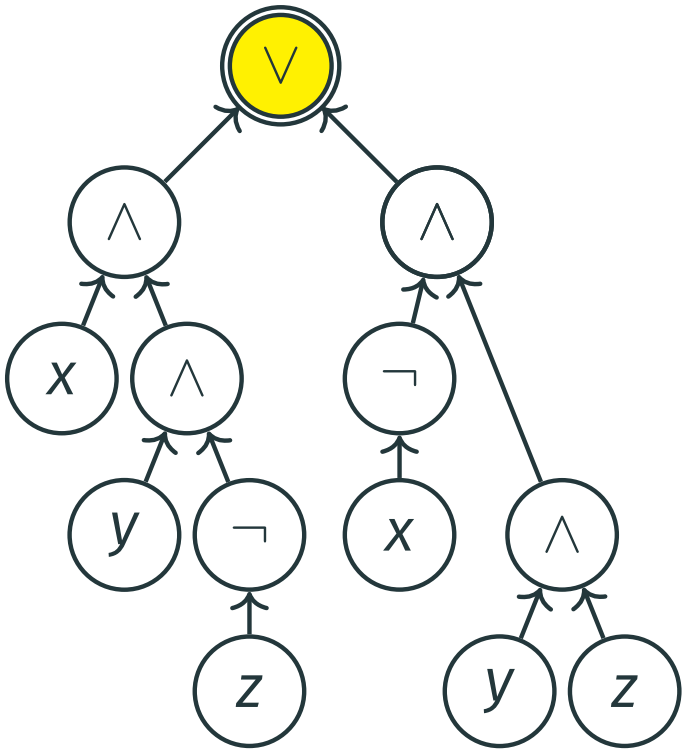
Example: what is this?



- NNF
- DNNF

(Circuit courtesy of Antoine Amarilli)

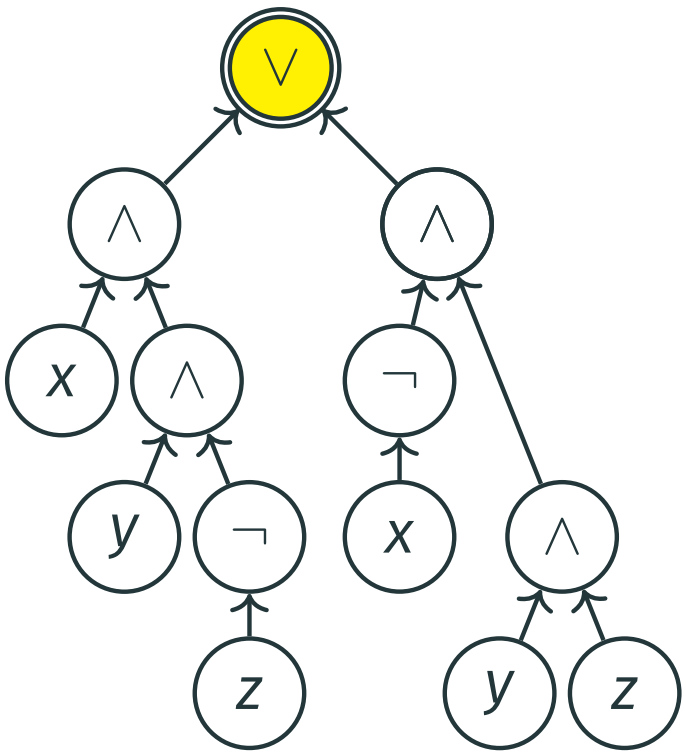
Example: what is this?



- NNF
- DNNF

(Circuit courtesy of Antoine Amarilli)

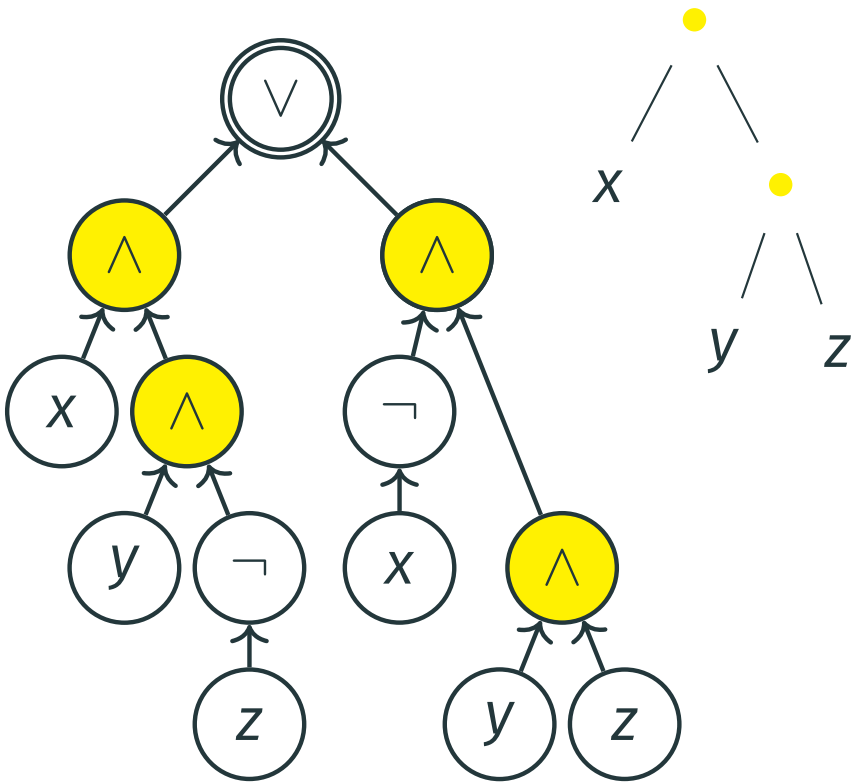
Example: what is this?



- NNF
- DNNF
- d-DNNF

(Circuit courtesy of Antoine Amarilli)

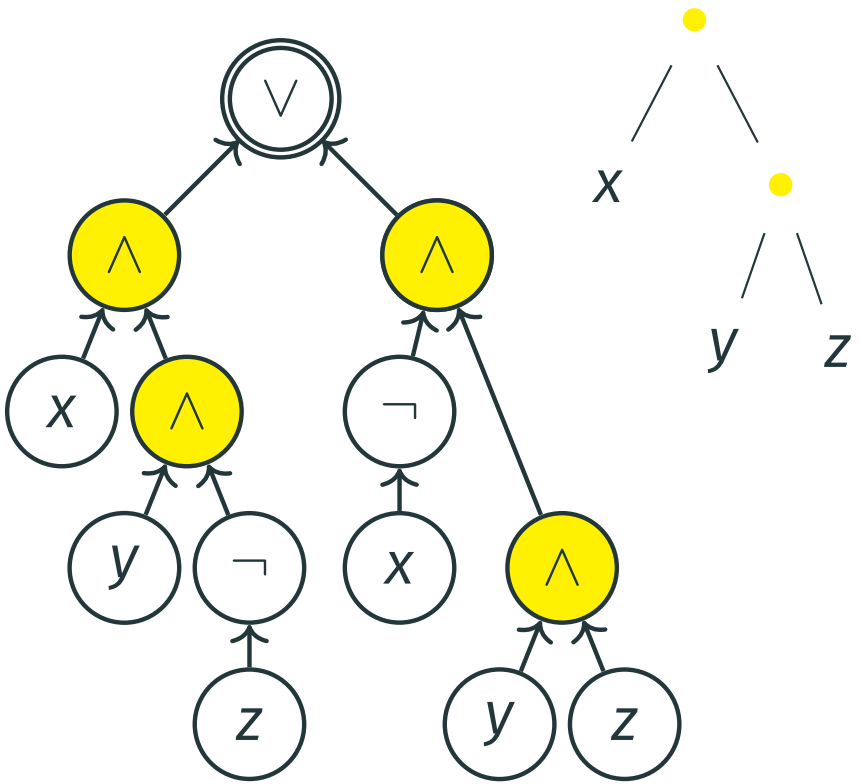
Example: what is this?



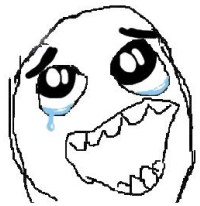
- NNF
- DNNF
- d-DNNF

(Circuit courtesy of Antoine Amarilli)

Example: what is this?



- NNF
- DNNF
- d-DNNF
- d-SDNNF



(Circuit courtesy of Antoine Amarilli)

Application to Probabilistic Databases

Complexity results

- Existing dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{I}_{\text{all}}$ is all instances
  - There is a class $\mathcal{S} \subseteq$ UCQs of safe queries

  $\rightarrow$ $q \in \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is PTIME

  $\rightarrow$ $q \in \text{UCQs} \setminus \mathcal{S} \implies \text{PQE}(q, \mathcal{I}_{\text{all}})$ is **#P-hard**

- Existing dichotomy result on instances
  - Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
  - $q \in$ MSO

  $\rightarrow$ $\text{PQE}(q, \mathcal{I}_k)$ has linear time complexity [Amarilli, Bourhis, & Senellart, 2015]

  $\rightarrow$ There is an FO query q_{hard} for which $\text{PQE}(q_{\text{hard}}, \mathcal{I})$ is **#P-hard** on any unbounded-treewidth graph family $\mathcal{I}$ (under some assumptions) [Amarilli, Bourhis, & Senellart, 2016]

On All Instances

- $\mathcal{I}_{\text{all}}$ is all instances

On All Instances

- $\mathcal{I}_{\text{all}}$ is all instances
- There is a class $\mathcal{IF} \subseteq$ UCQs of inversion free queries

On All Instances

- $\mathcal{I}_{\text{all}}$ is all instances
- There is a class $\mathcal{IF} \subseteq$ UCQs of inversion free queries

$\rightarrow$ $q \in \mathcal{IF} \iff$ we can compute in PTIME $\mathsf{Prov}(q, I)$ as an OBDD (hence in particular $\mathcal{IF} \subseteq$ safe UCQs)

On All Instances

- $\mathcal{I}_{\text{all}}$ is all instances
- There is a class $\mathcal{IF} \subseteq$ UCQs of inversion free queries

$\rightarrow$ $q \in \mathcal{IF} \iff$ we can compute in PTIME $\mathsf{Prov}(q, I)$ as an OBDD (hence in particular $\mathcal{IF} \subseteq$ safe UCQs)

Open Characterisation for FBDDs? (we know $\mathcal{IF} \subsetneq ??? \subsetneq$ safe UCQs)

On All Instances

- $\mathcal{I}_{\text{all}}$ is all instances
- There is a class $\mathcal{IF} \subseteq$ UCQs of inversion free queries

$\rightarrow$ $q \in \mathcal{IF} \iff$ we can compute in PTIME $\mathsf{Prov}(q, I)$ as an OBDD (hence in particular $\mathcal{IF} \subseteq$ safe UCQs)

Open Characterisation for FBDDs? (we know $\mathcal{IF} \subsetneq ??? \subsetneq$ safe UCQs)

Open Can we tractably compute the provenance of safe UCQs as d-DNNFs?

On Bounded Treewidth Instances

- Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
- Fix $q \in$ MSO

On Bounded Treewidth Instances

- Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
- Fix $q \in$ MSO

Theorem (Amarilli, Bourhis, & Senellart, 2015)
Given $I \in \mathcal{I}_k$, we can compute in linear time a Boolean circuit $C_{q,I}$ capturing $\mathsf{Prov}(q, I)$. *Moreover $C_{q,I}$ has bounded treewidth (i.e., $f(|q|, k)$ for some function f)*

On Bounded Treewidth Instances

- Fix $k \in \mathbb{N}$. $\mathcal{I}_k =$ all instances of treewidth $\leqslant k$
- Fix $q \in$ MSO

Theorem (Amarilli, Bourhis, & Senellart, 2015)
Given $I \in \mathcal{I}_k$, we can compute in linear time a Boolean circuit $C_{q,I}$ capturing $\mathsf{Prov}(q, I)$. *Moreover $C_{q,I}$ has bounded treewidth (i.e., $f(|q|, k)$ for some function f)*

- Nice, how do I compute the probability?

Re-explaining Message-Passing

Theorem (Lauritzen, & Spielgelhalter, 1988)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth $\leqslant k$, we can compute its probability in time $O(f(k) \times |C|)$, where f is singly exponential

Re-explaining Message-Passing

Theorem (Lauritzen, & Spielgelhalter, 1988)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth $\leqslant k$, we can compute its probability in time $O(f(k) \times |C|)$, where f is singly exponential

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth k, we can compute in linear time a d-SDNNF equivalent to C (hence its probability) in time $O(f(k) \times |C|)$, where f is singly exponential

Re-explaining Message-Passing

Theorem (Lauritzen, & Spielgelhalter, 1988)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth $\leqslant k$, we can compute its probability in time $O(f(k) \times |C|)$, where f is singly exponential

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth k, we can compute in linear time a d-SDNNF equivalent to C (hence its probability) in time $O(f(k) \times |C|)$, where f is singly exponential

- Bounded treewidth circuits $\xrightarrow{\text{linear time}}$ d-SDNNFs

Re-explaining Message-Passing

Theorem (Lauritzen, & Spielgelhalter, 1988)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth $\leqslant k$, we can compute its probability in time $O(f(k) \times |C|)$, where f is singly exponential

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth k, we can compute in linear time a d-SDNNF equivalent to C (hence its probability) in time $O(f(k) \times |C|)$, where f is singly exponential

- Bounded treewidth circuits $\xrightarrow{\text{linear time}}$ d-SDNNFs
- Consequences for enumeration

Lower Bounds

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)

Let φ be a monotone DNF, let $a := \mathrm{arity}(\varphi)$ and $d := \mathrm{degree}(\varphi)$. Then any d-SDNNF for φ has size $\geqslant 2^{\left\lfloor \frac{\mathrm{tw}(\varphi)}{3 \times a^3 \times d^2} \right\rfloor} - 1$

Lower Bounds

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)

Let φ be a monotone DNF, let $a := \mathrm{arity}(\varphi)$ *and* $d := \mathrm{degree}(\varphi)$. *Then any d-SDNNF for φ has size* $\geqslant 2^{\left\lfloor \frac{\mathrm{tw}(\varphi)}{3 \times a^3 \times d^2} \right\rfloor} - 1$

$\rightarrow$ Consequence for probabilistic databases:

Lower Bounds

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)
Let φ be a monotone DNF, let $a := \mathrm{arity}(\varphi)$ and $d := \mathrm{degree}(\varphi)$. Then any d-SDNNF for φ has size $\geqslant 2^{\left\lfloor \frac{\mathrm{tw}(\varphi)}{3 \times a^3 \times d^2} \right\rfloor} - 1$

$\rightarrow$ Consequence for probabilistic databases:

Theorem (Amarilli, Monet, & Senellart, hopefuly 2018)
There is a constant $d \in \mathbb{N}$ such that the following is true. Let σ be an arity-2 signature, and Q a connected $\mathrm{UCQ}^{\neq}$ which is intricate on σ. For any instance I on σ, any d-SDNNF representing the lineage of Q on I has size $2^{\Omega(\mathrm{tw}(I)^{1/d})}$

Conclusion

- Knowledge Compilation as a tool for probabilistic databases and provenance computation

Conclusion

- Knowledge Compilation as a tool for probabilistic databases and provenance computation
- Upper bounds in KC $\Longrightarrow$ uppers bounds for PQE

Conclusion

- Knowledge Compilation as a tool for probabilistic databases and provenance computation
- Upper bounds in KC $\Longrightarrow$ uppers bounds for PQE
- Lower bounds in KC $\Longrightarrow$ limits of the intensional approach of PQE

Conclusion

- Knowledge Compilation as a tool for probabilistic databases and provenance computation
- Upper bounds in KC $\Longrightarrow$ uppers bounds for PQE
- Lower bounds in KC $\Longrightarrow$ limits of the intensional approach of PQE

Thanks for your attention!