First-Order Knowledge Compilation

Guy Van den Broeck

UCLA

Dagstuhl

Sept 18, 2017

STAR AI RESEARCH LAB UCLA

Overview

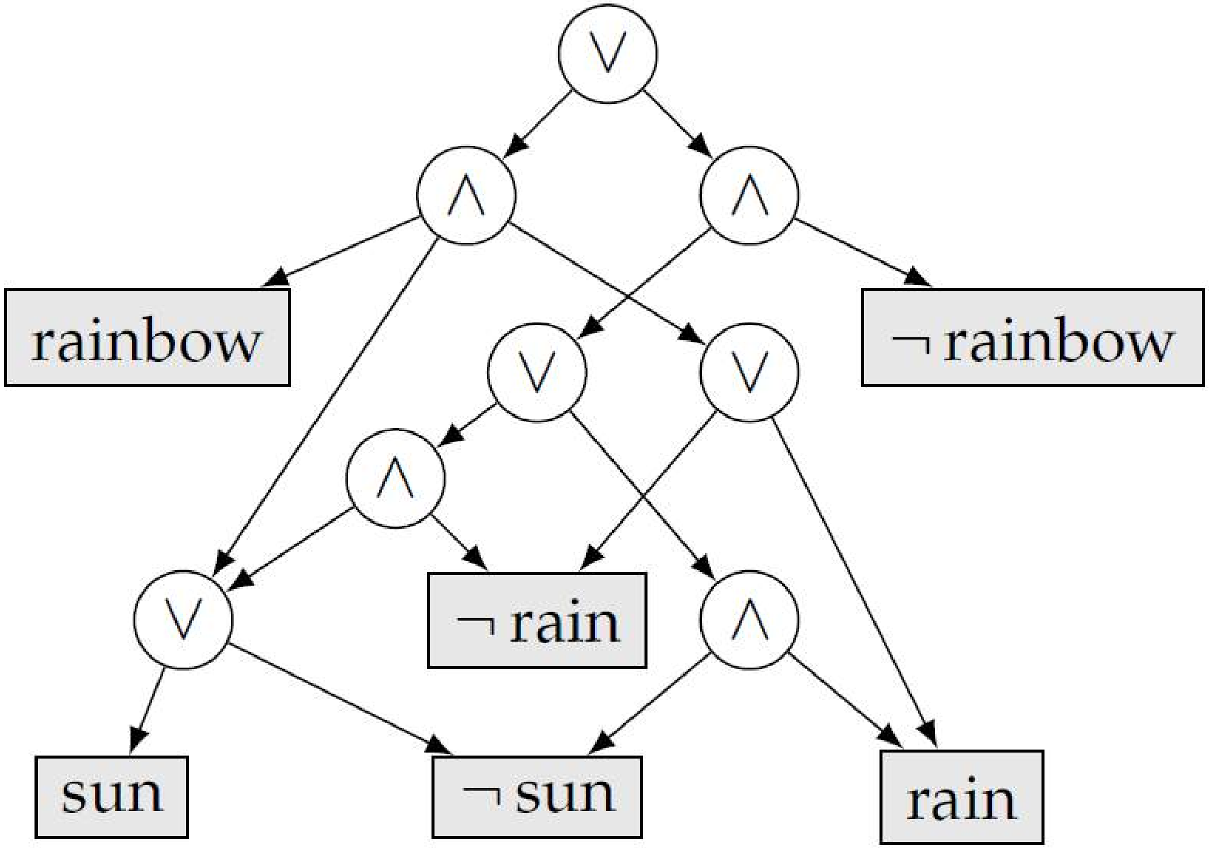
1. Propositional Refresher
2. Primer: A First-Order Tractable Language
3. Probabilistic Databases
4. Symmetric First-Order Model Counting
5. Lots of Pointers

Overview

1. **Propositional Refresher**
2. Primer: A First-Order Tractable Language
3. Probabilistic Databases
4. Symmetric First-Order Model Counting
5. Lots of Pointers

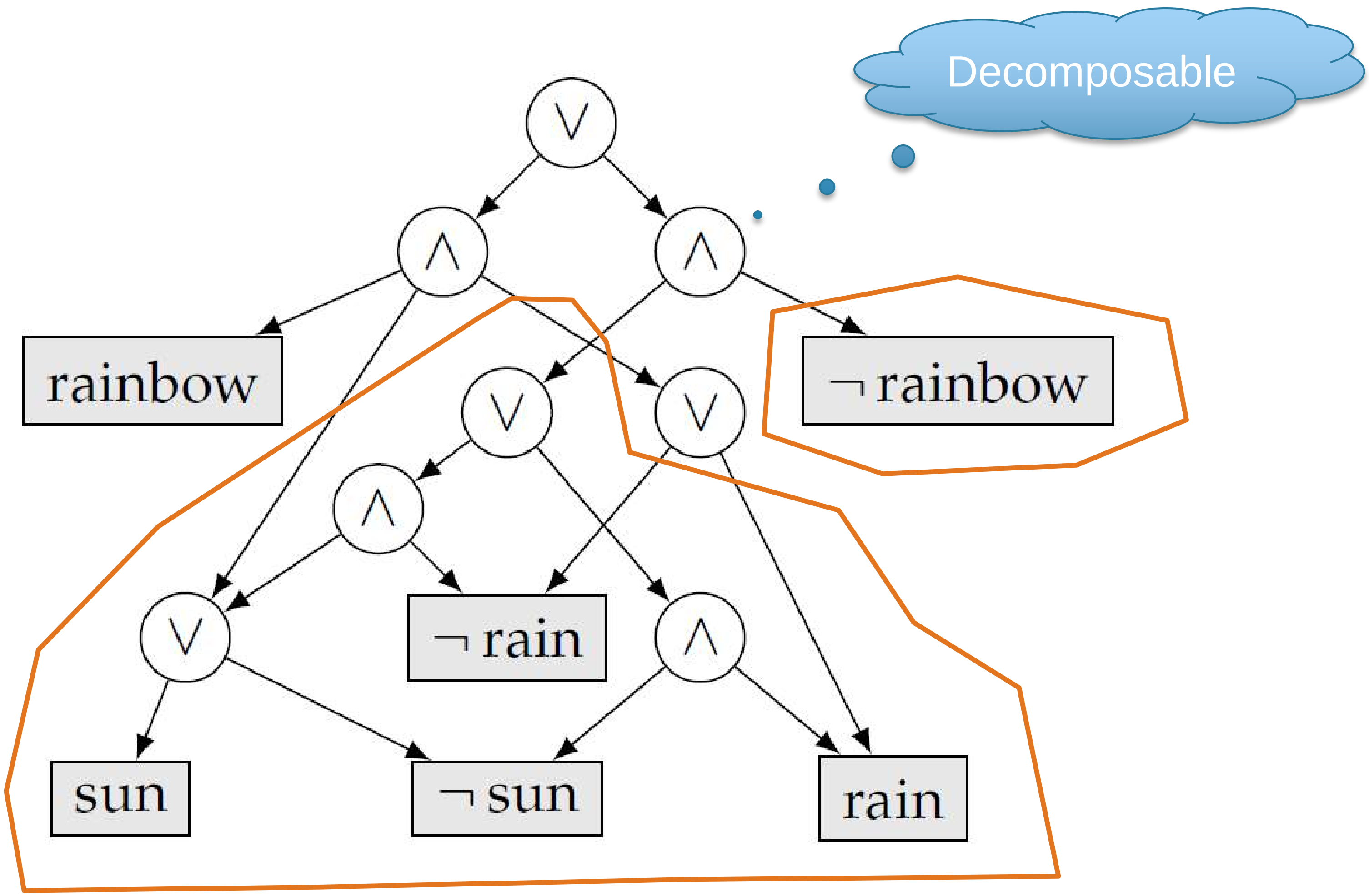
Negation Normal Form

Δ = (sun ∧ rain ⇒ rainbow)



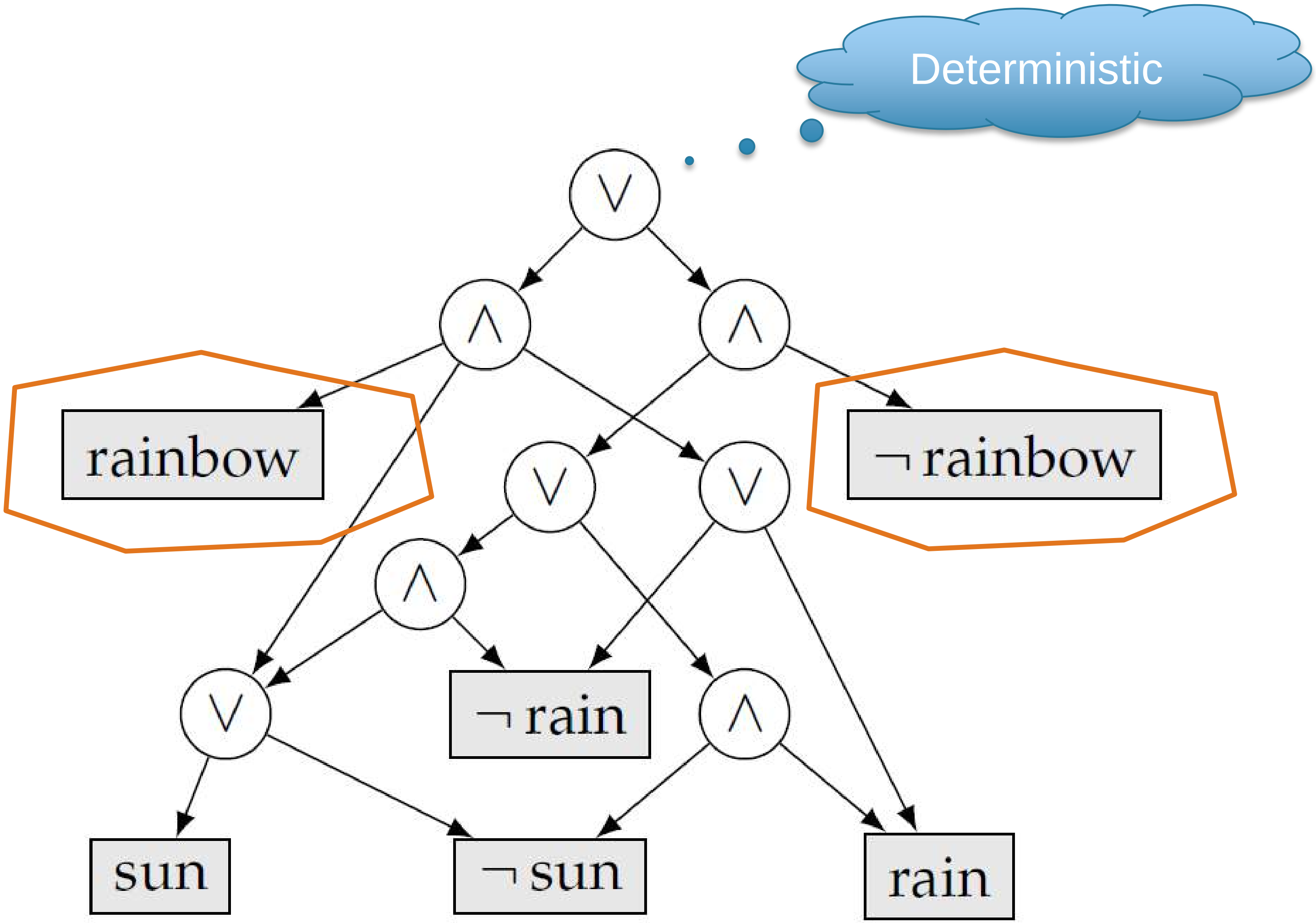
[Darwiche 2002]

Decomposable NNF



[Darwiche 2002]

Deterministic NNF



[Darwiche 2002]

Model Counting

- Model = solution to a propositional logic formula Δ
- Model counting = #SAT

Δ = (Rain ⇒ Cloudy)

Rain	Cloudy	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

+

#SAT = 3

Model Counting

- Model = solution to a propositional logic formula Δ
- Model counting = #SAT

Δ = (Rain ⇒ Cloudy)

Rain	Cloudy	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

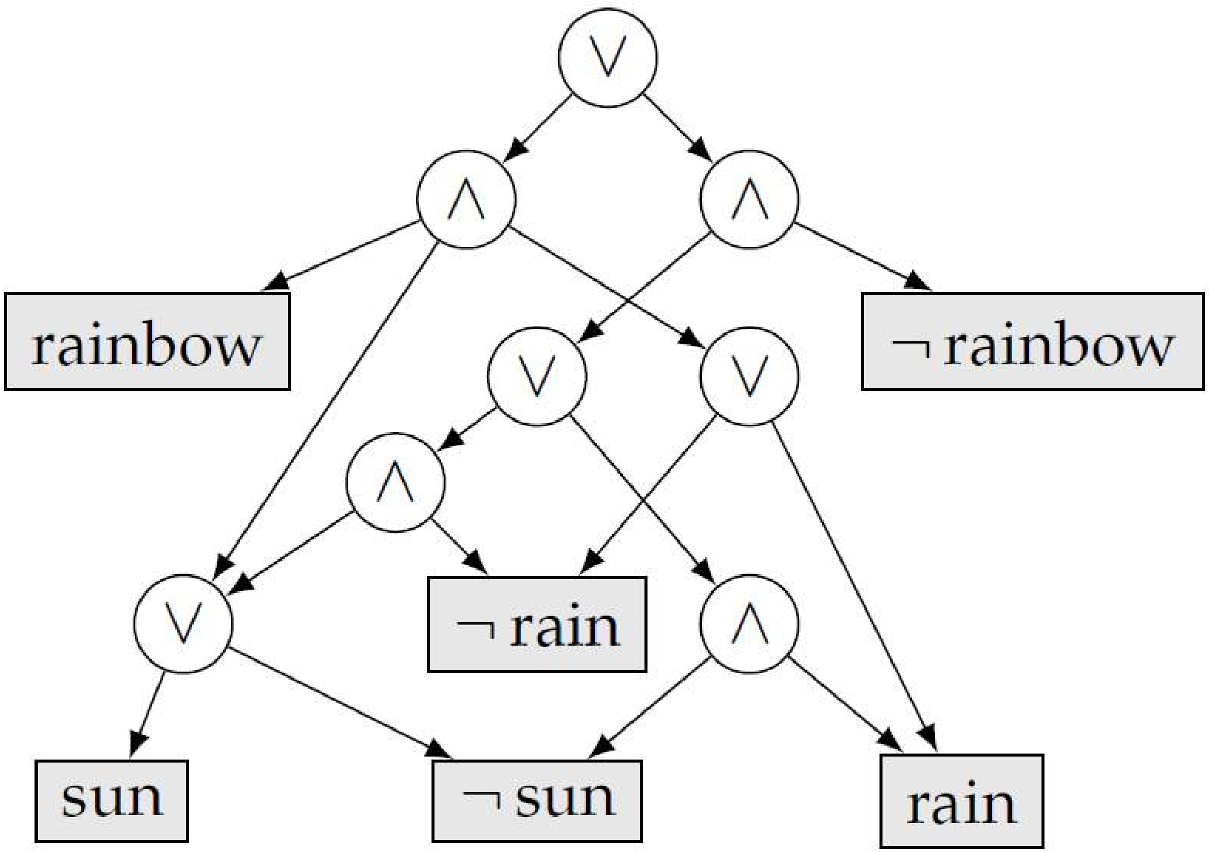
+ ______

#SAT = 3

[Valiant] #P-hard, even for 2CNF

Deterministic Decomposable NNF

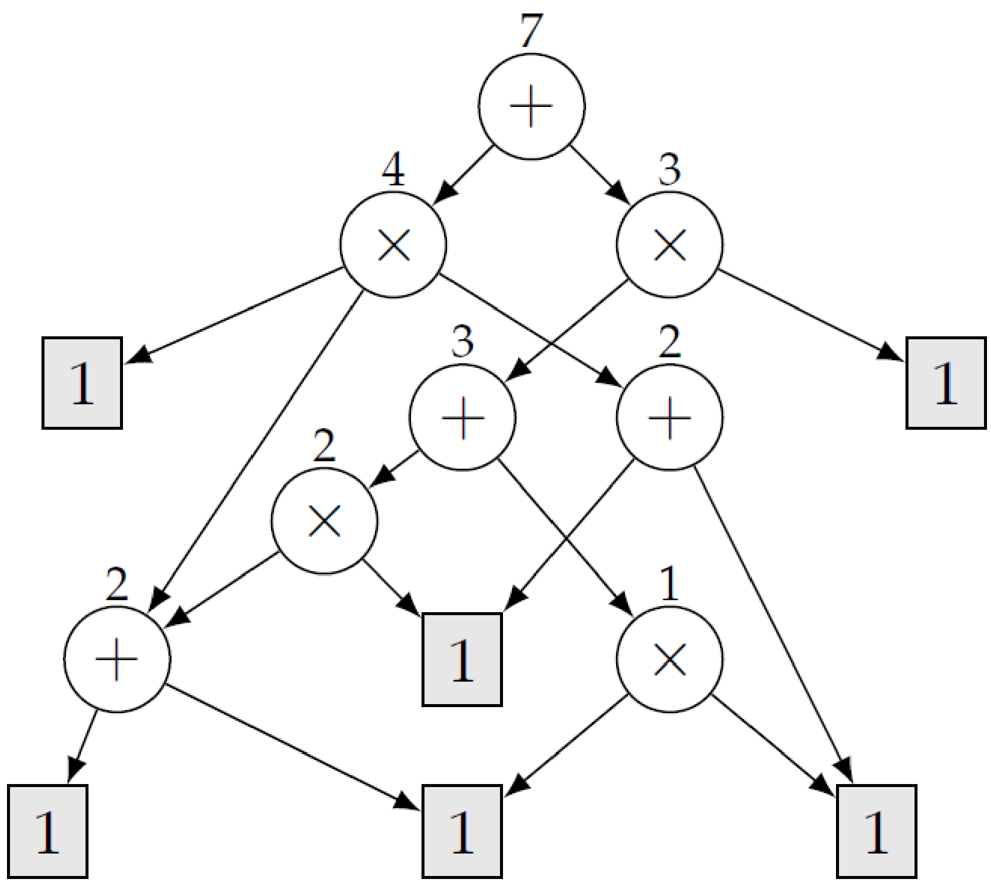
Model Counting?



[Darwiche 2002]

Deterministic Decomposable NNF

Model Counting



[Darwiche 2002]

Weighted Model Count

- Weights for assignments to variables
- Model weight = product of variable weights

Δ = (Rain ⇒ Cloudy)

Rain	Cloudy	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

Weighted Model Count

- Weights for assignments to variables
- Model weight = product of variable weights

Δ = (Rain ⇒ Cloudy)

Rain

w(R)	w(¬R)
1	2

Cloudy

w(C)	w(¬C)
3	5

Rain	Cloudy	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

Weighted Model Count

- Weights for assignments to variables
- Model weight = product of variable weights

Δ = (Rain ⇒ Cloudy)

Rain

w(R)	w(¬R)
1	2

Cloudy

w(C)	w(¬C)
3	5

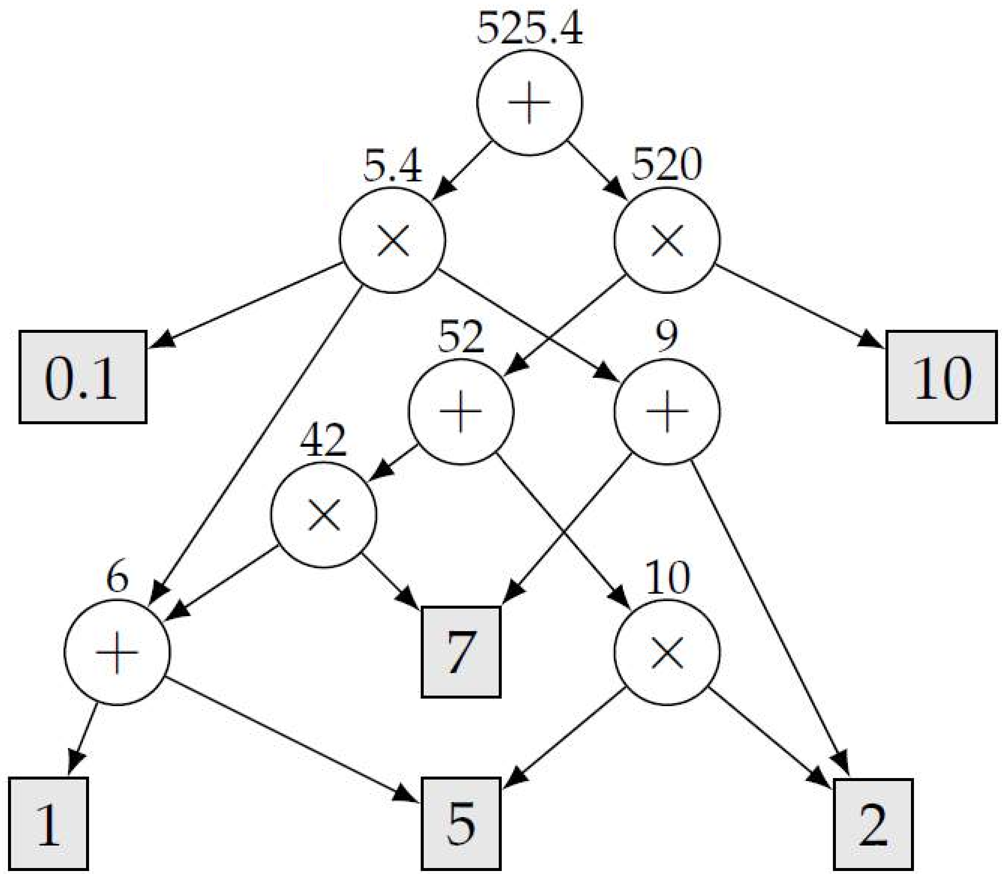
Rain	Cloudy	Model?	Weight
T	T	Yes	1 * 3 = 3
T	F	No	0
F	T	Yes	2 * 3 = 6
F	F	Yes	2 * 5 = 10

+

WMC = 19

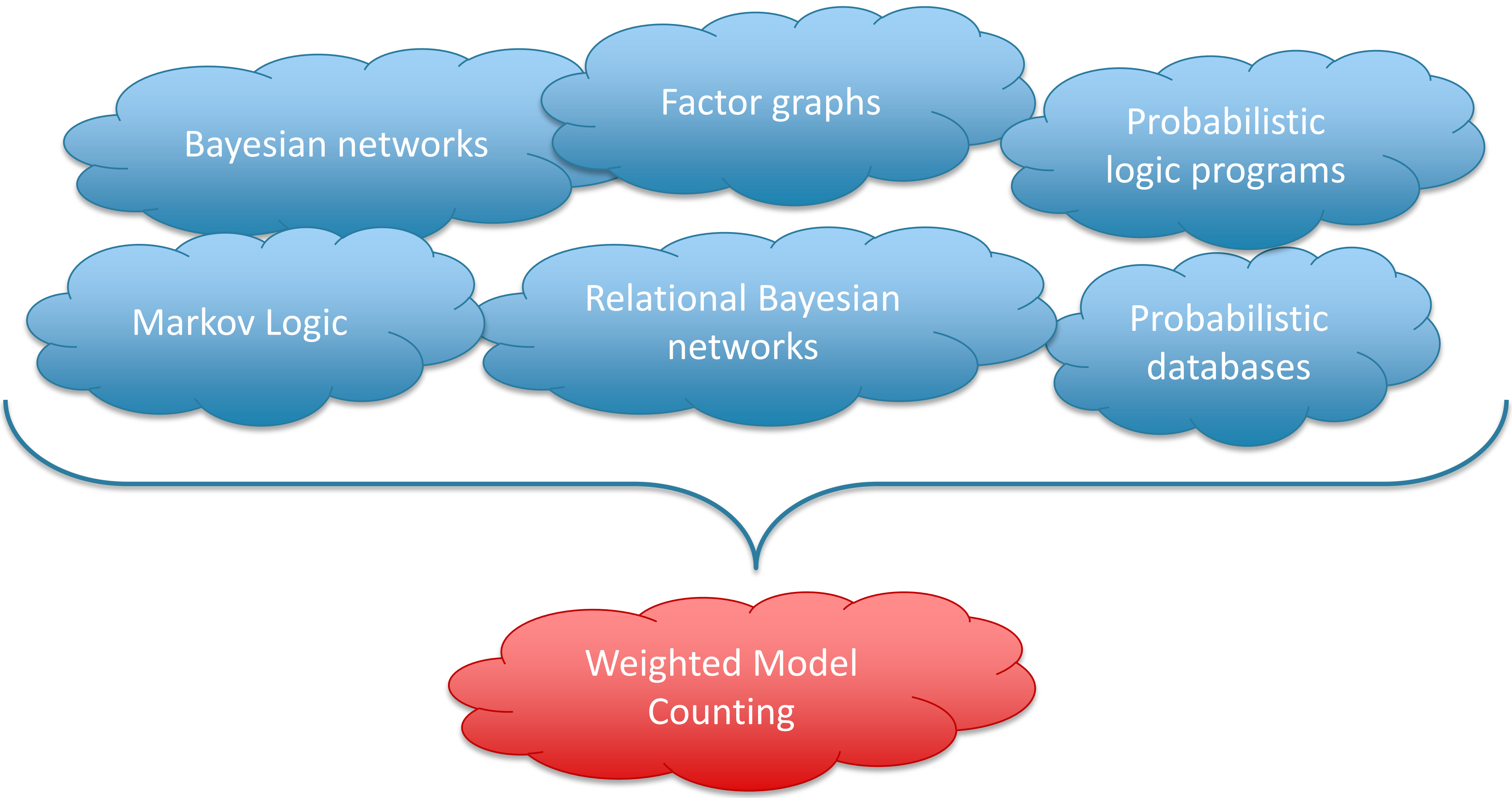
Deterministic Decomposable NNF

Weighted Model Counting



[Darwiche 2002]

Assembly language for probabilistic reasoning



[Chavira 2006, Chavira 2008, Sang 2005, Fierens 2015]

Probability of a Sentence

- Special case of WMC
- Weights are probabilities: w(R) + w(¬R) = 1

Δ = (Rain ⇒ Cloudy)

Rain

w(R)	w(¬R)
0.8	0.2

Cloudy

w(C)	w(¬C)
0.5	0.5

Rain	Cloudy	Model?	Weight
T	T	Yes	.8 * .5 = .4
T	F	No	0
F	T	Yes	.2 * 5 = .1
F	F	Yes	.2 * 5 = .1

- Simplifies some details (smoothing)

Probability of a Sentence

- Special case of WMC
- Weights are probabilities: w(R) + w(¬R) = 1

Δ = (Rain ⇒ Cloudy)

Rain

w(R)	w(¬R)
0.8	0.2

Cloudy

w(C)	w(¬C)
0.5	0.5

Rain	Cloudy	Model?	Weight
T	T	Yes	.8 * .5 = .4
T	F	No	0
F	T	Yes	.2 * 5 = .1
F	F	Yes	.2 * 5 = .1

+

P(Δ) = 0.6

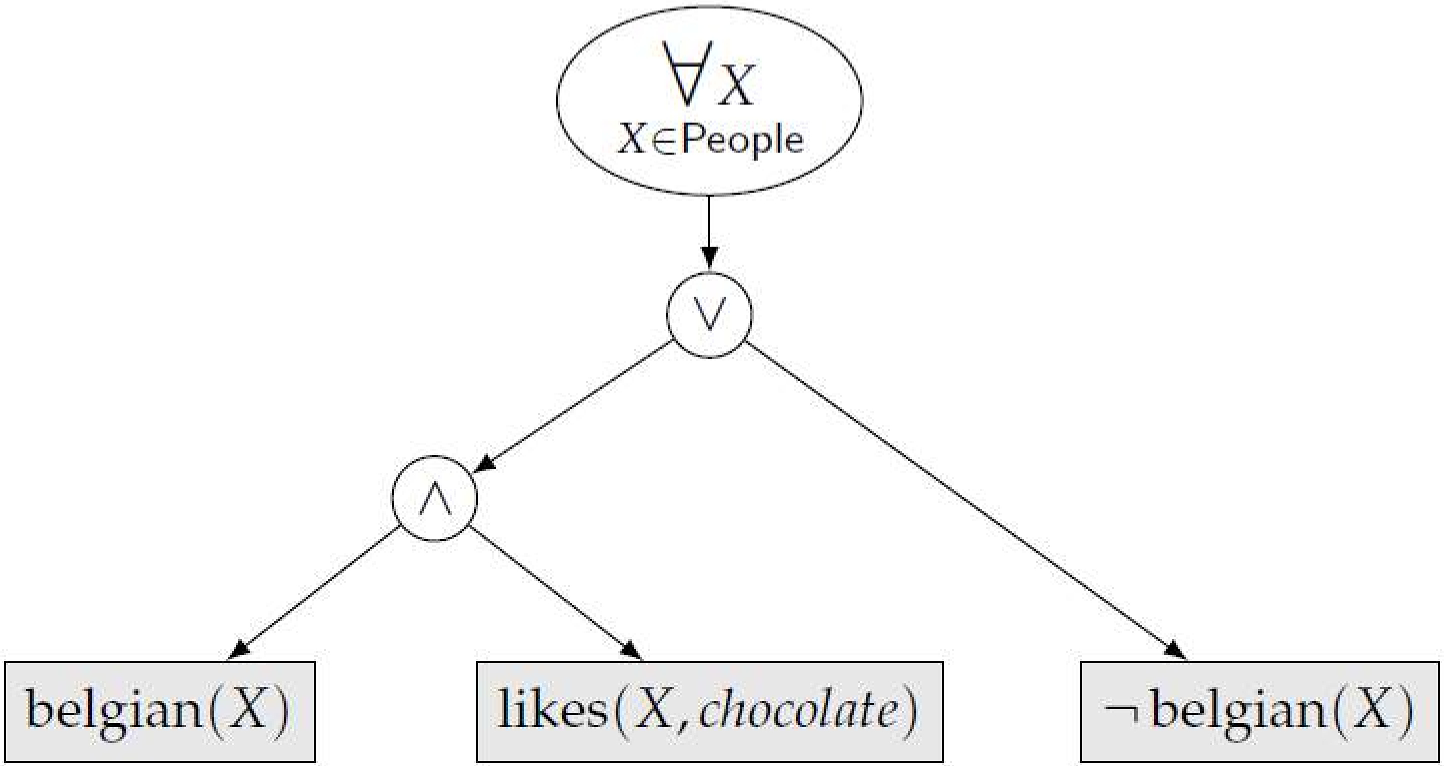
- Simplifies some details (smoothing)

Overview

1. Propositional Refresher
2. **Primer: A First-Order Tractable Language**
3. Probabilistic Databases
4. Symmetric First-Order Model Counting
5. Lots of Pointers

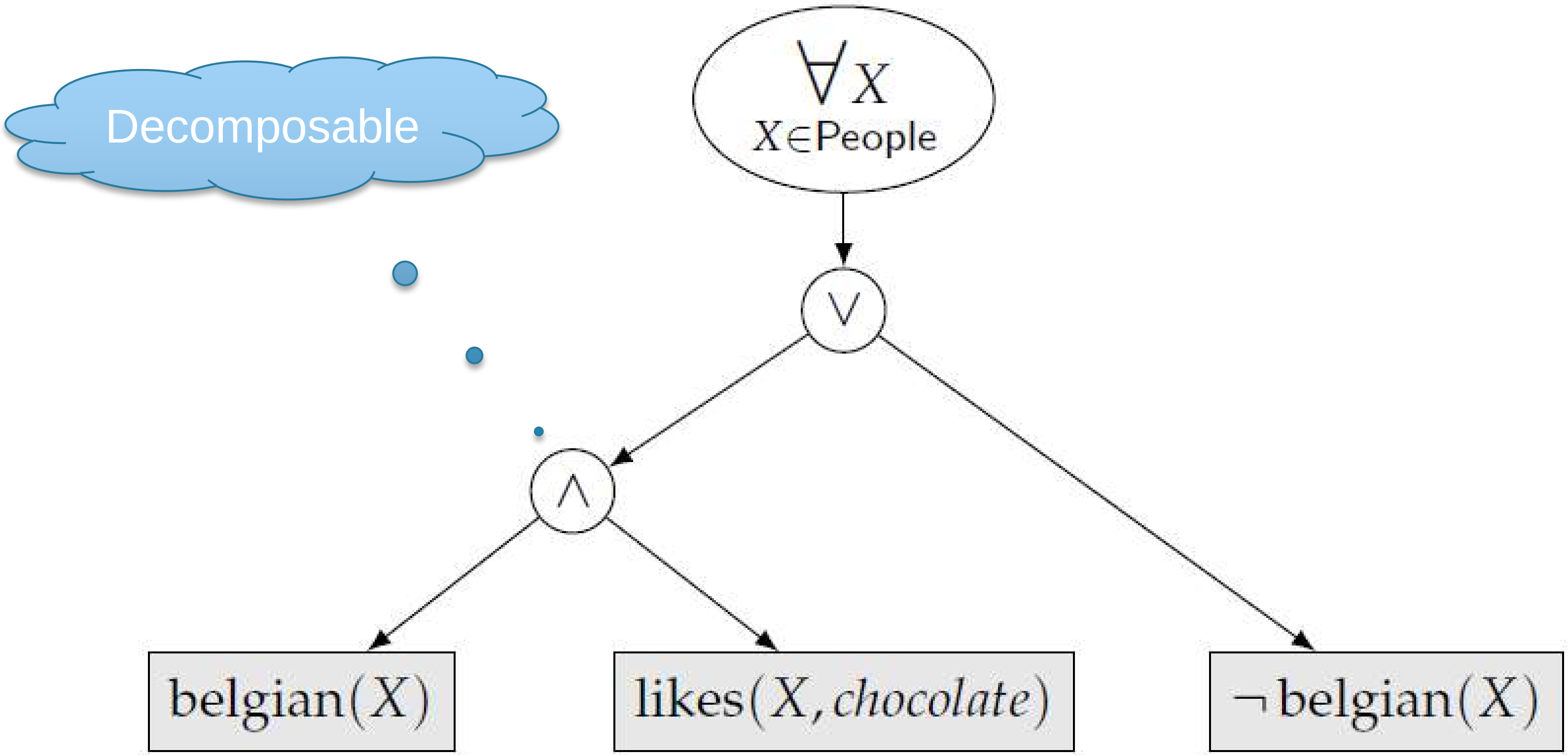
First-Order NNF

$$\forall X, X \in \text{People} : \text{belgian}(X) \Rightarrow \text{likes}(X, \textit{chocolate})$$



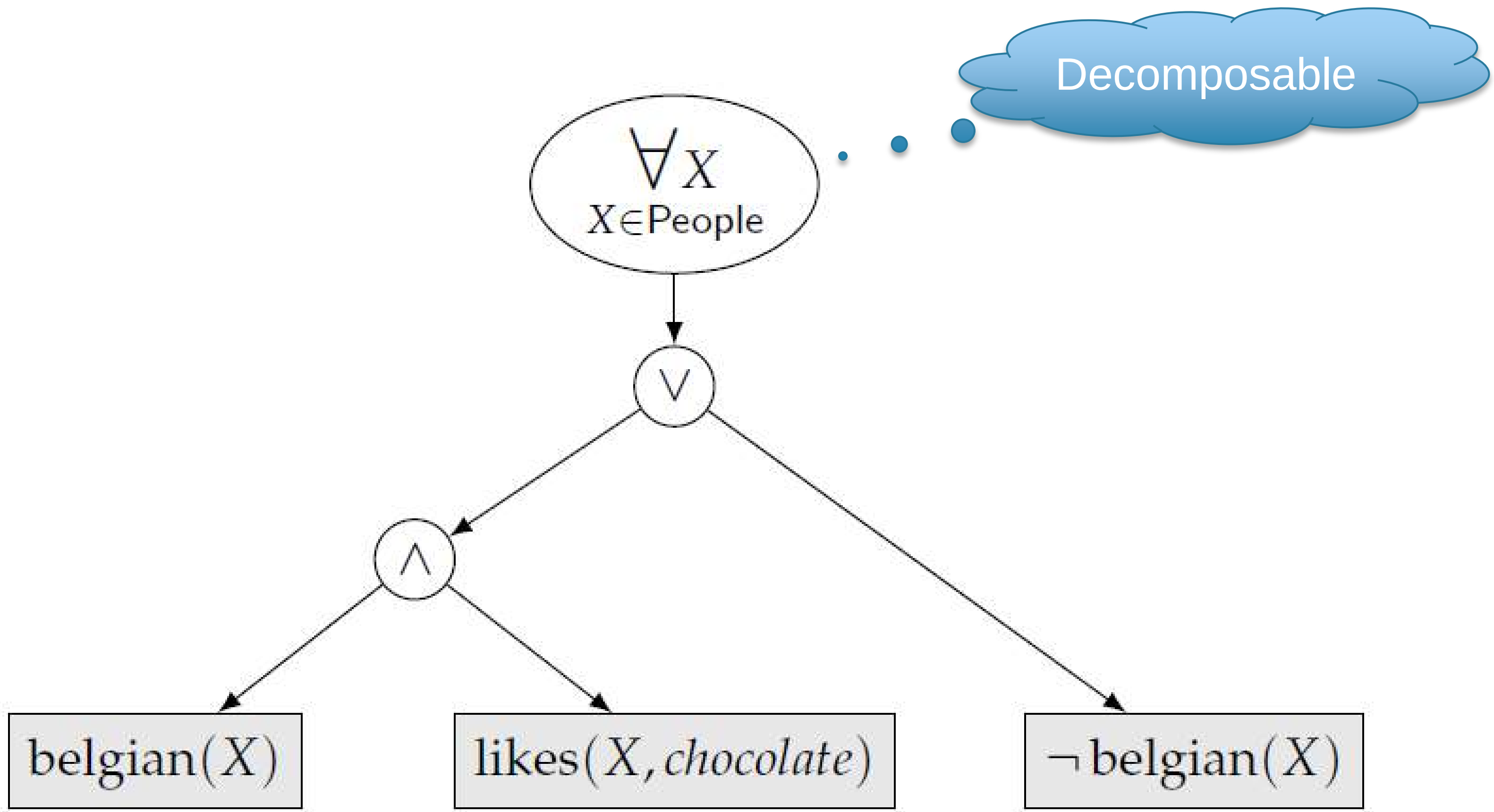
[Van den Broeck 2013]

First-Order Decomposability



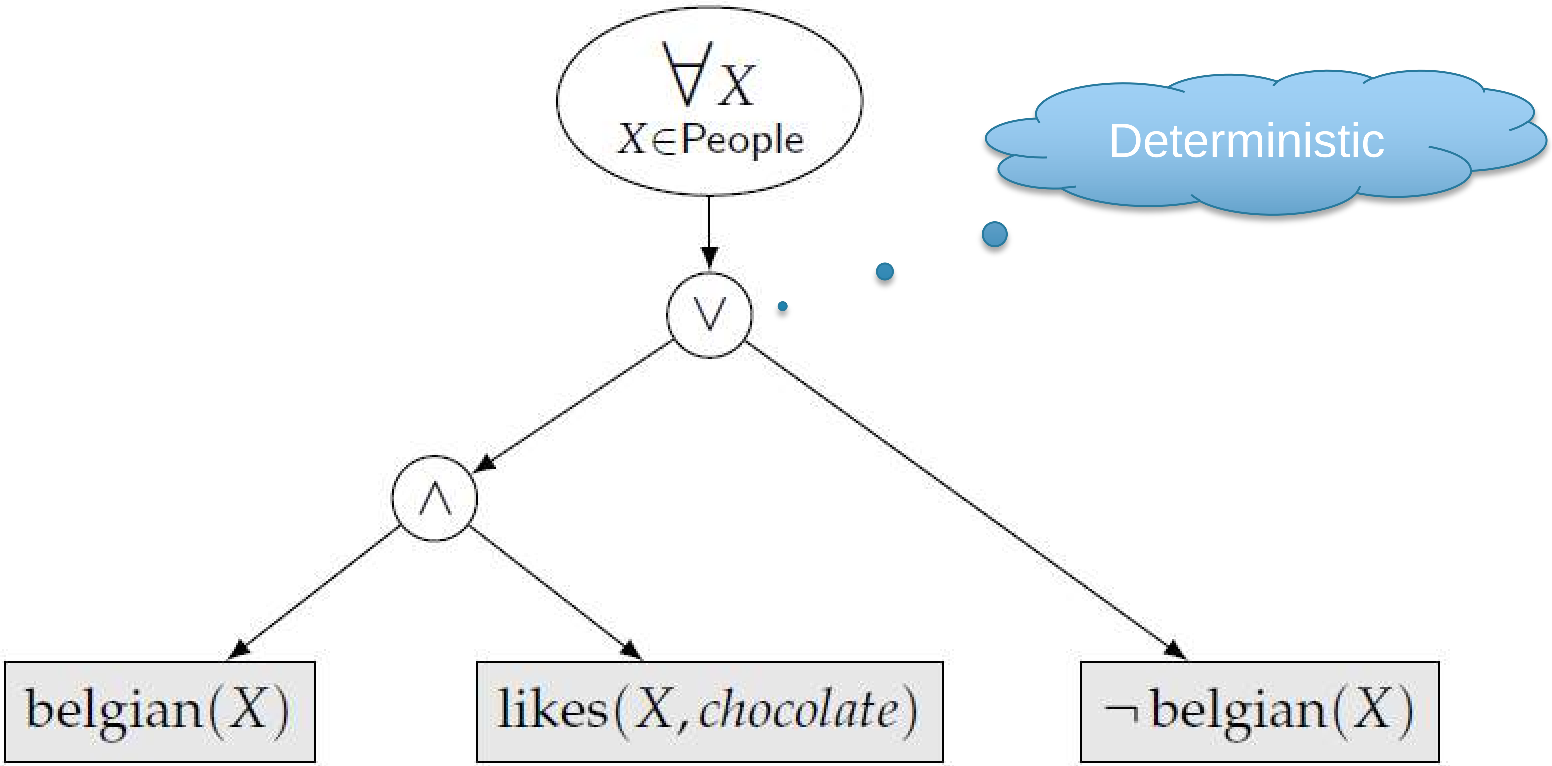
[Van den Broeck 2013]

First-Order Decomposability



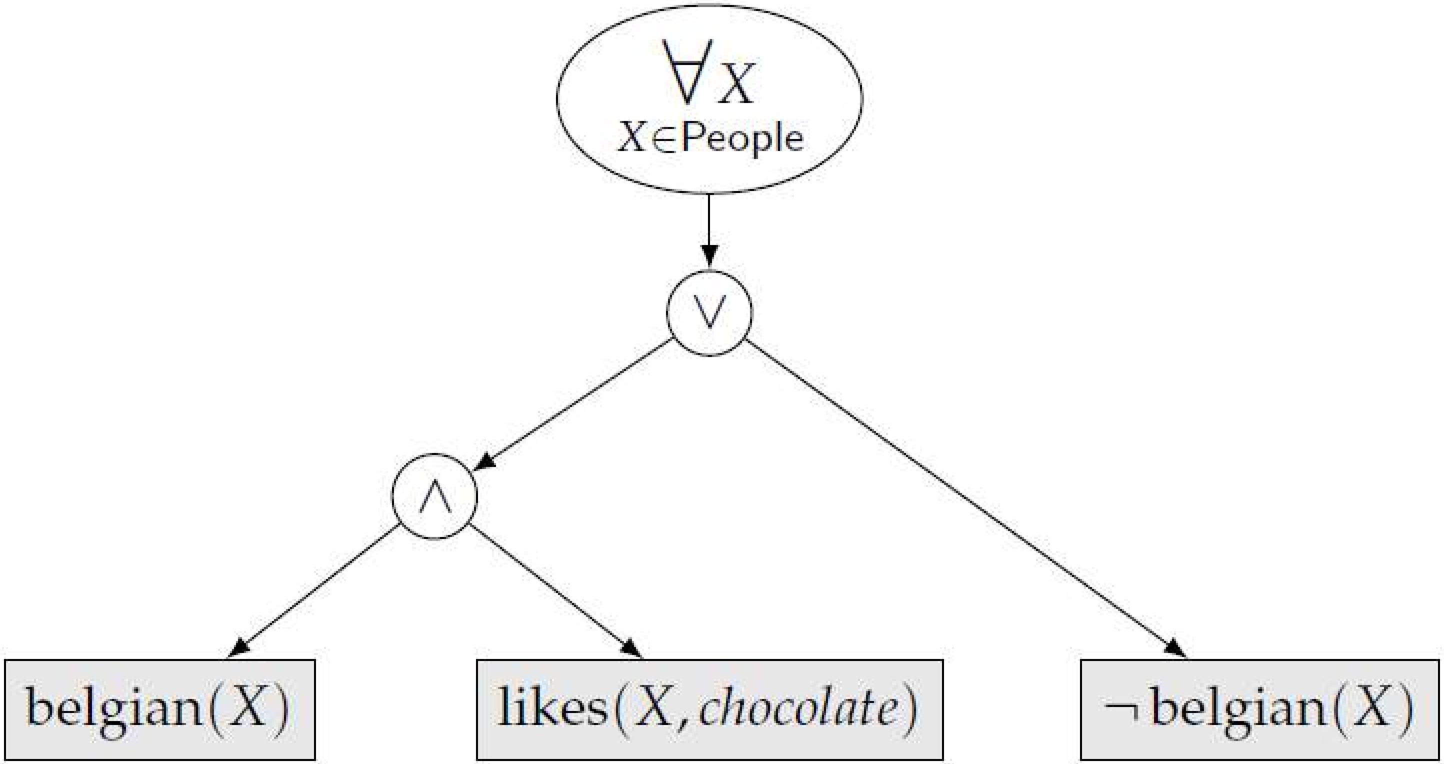
[Van den Broeck 2013]

First-Order Determinism



[Van den Broeck 2013]

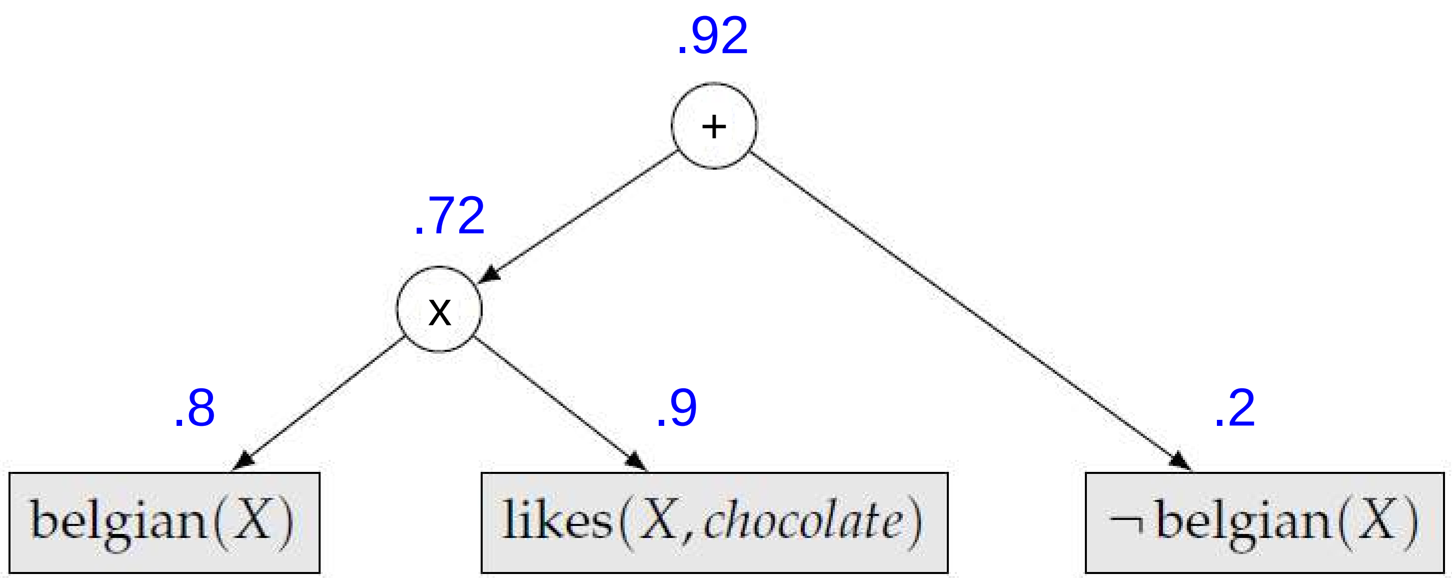
Probability of Sentence (WMC)



[Van den Broeck 2013]

Probability of Sentence (WMC)

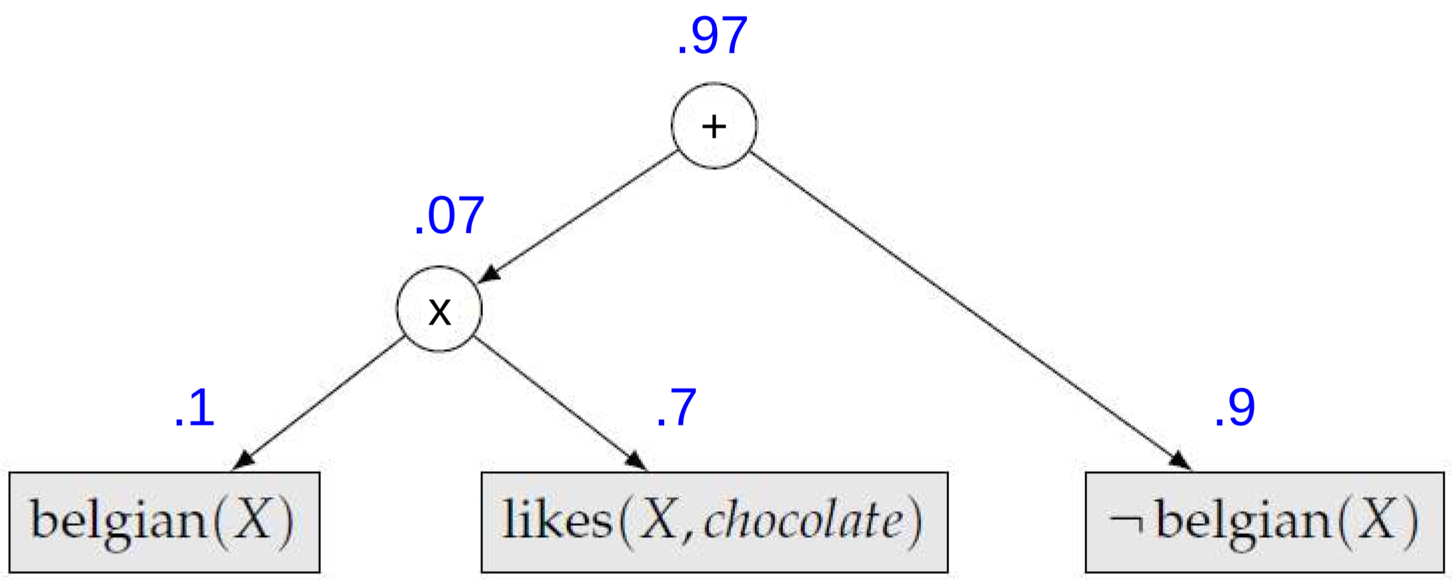
For X = guy: .92



[Van den Broeck 2013]

Probability of Sentence (WMC)

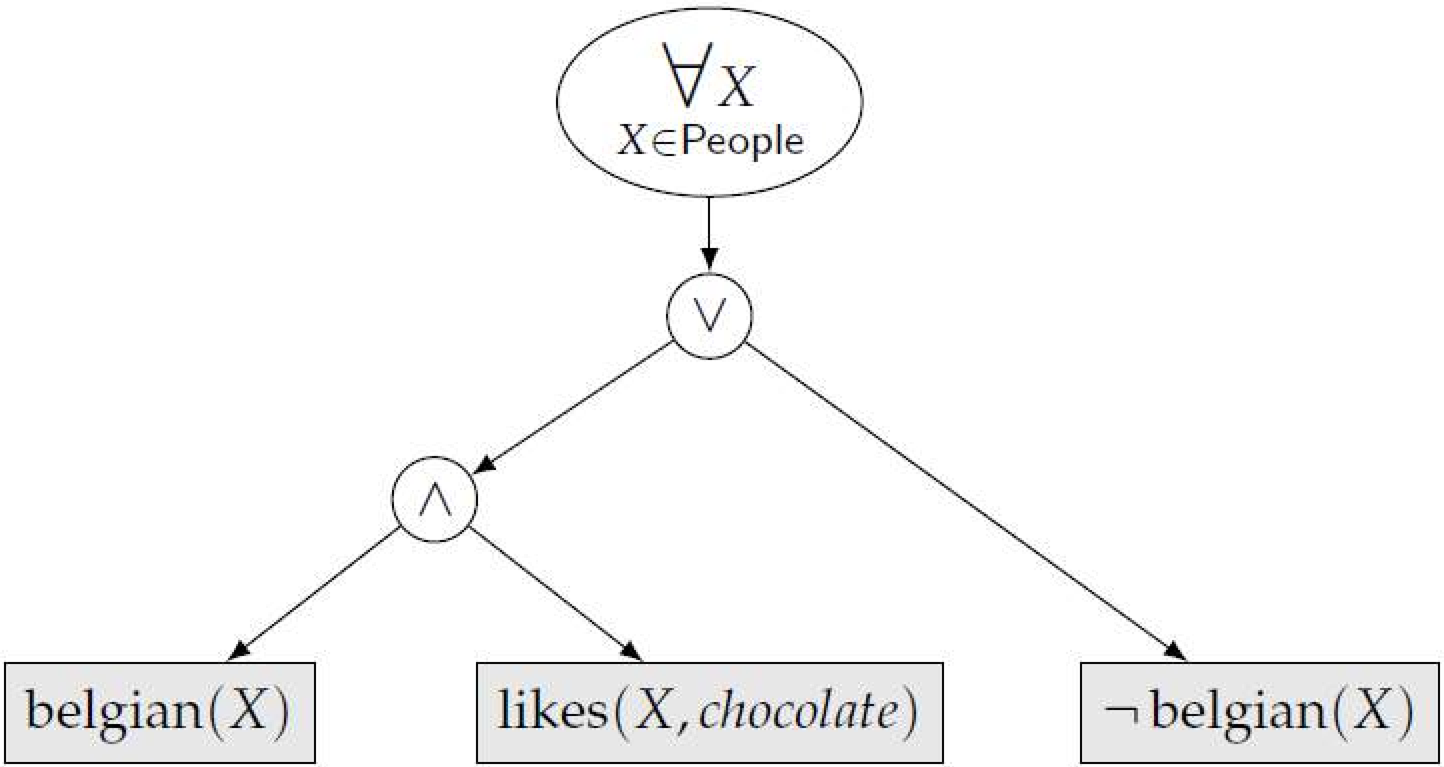
For X = guy: .92
X = mary: .97



[Van den Broeck 2013]

Probability of Sentence (WMC)

For all people:
.92 x .97 = .89



[Van den Broeck 2013]

Evaluate Probability on FO Circuit*

** Also non-NNF to simplify examples. Some rules redundant given others.*

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$

Negation

** Also non-NNF to simplify examples. Some rules redundant given others.*

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$

Negation

$P(Q1 \wedge Q2) = P(Q1)\, P(Q2)$
$P(Q1 \vee Q2) = 1 - (1 - P(Q1))\, (1 - P(Q2))$

Decomposable ∧,∨

** Also non-NNF to simplify examples. Some rules redundant given others.*

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$

Negation

$P(Q1 \wedge Q2) = P(Q1)\, P(Q2)$
$P(Q1 \vee Q2) = 1 - (1 - P(Q1))\, (1 - P(Q2))$

Decomposable ∧,∨

$P(\forall z\; Q) = \Pi_{A \in Domain}\; P(Q[A/z])$
$P(\exists z\; Q) = 1 - \Pi_{A \in Domain}\; (1 - P(Q[A/z]))$

Decomposable ∀,∃

** Also non-NNF to simplify examples. Some rules redundant given others.*

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$ — Negation

$P(Q1 \land Q2) = P(Q1)\, P(Q2)$
$P(Q1 \lor Q2) = 1 - (1 - P(Q1))\, (1 - P(Q2))$ — Decomposable ∧,∨

$P(\forall z\, Q) = \Pi_{A \in Domain}\ P(Q[A/z])$
$P(\exists z\, Q) = 1 - \Pi_{A \in Domain}\ (1 - P(Q[A/z]))$ — Decomposable ∀,∃

$P(Q1 \land Q2) = P(Q1) + P(Q2) - 1$
$P(Q1 \lor Q2) = P(Q1) + P(Q2)$ — Deterministic ∧,∨

** Also non-NNF to simplify examples. Some rules redundant given others.*

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$	Negation
$P(Q1 \wedge Q2) = P(Q1)\,P(Q2)$ $P(Q1 \vee Q2) = 1 - (1 - P(Q1))\,(1 - P(Q2))$	Decomposable ∧,∨
$P(\forall z\, Q) = \prod_{A \in Domain} P(Q[A/z])$ $P(\exists z\, Q) = 1 - \prod_{A \in Domain} (1 - P(Q[A/z]))$	Decomposable ∀,∃
$P(Q1 \wedge Q2) = P(Q1) + P(Q2) - 1$ $P(Q1 \vee Q2) = P(Q1) + P(Q2)$	Deterministic ∧,∨
$P(\forall z\, Q) = 1 - \sum_{A \in Domain} 1 - P(Q[A/z])$ $P(\exists z\, Q) = \sum_{A \in Domain} P(Q[A/z])$	Deterministic ∀,∃

** Also non-NNF to simplify examples. Some rules redundant given others.*

Limitations

$$H_0 = \forall x \forall y\ \mathrm{Smoker}(x) \lor \mathrm{Friend}(x,y) \lor \mathrm{Jogger}(y)$$

The decomposable ∀-rule: $P(\forall z\ Q) = \prod_{A \in \mathrm{Domain}} P(Q[A/z])$

[Suciu'11]

Limitations

$H_0 = \forall x \forall y\ \text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

The decomposable ∀-rule: $P(\forall z\ Q) = \prod_{A \in \text{Domain}} P(Q[A/z])$

… does not apply:

$H_0[\text{Alice}/x]$ and $H_0[\text{Bob}/x]$ are dependent:

Dependent

$\forall y\ (\text{Smoker}(\text{Alice}) \vee \text{Friend}(\text{Alice},y) \vee \text{Jogger}(y))$

$\forall y\ (\text{Smoker}(\text{Bob}) \vee \text{Friend}(\text{Bob},y) \vee \text{Jogger}(y))$

[Suciu'11]

Limitations

$H_0 = \forall x \forall y\ Smoker(x) \lor Friend(x,y) \lor Jogger(y)$

The decomposable ∀-rule: $P(\forall z\ Q) = \Pi_{A \in Domain}\ P(Q[A/z])$

… does not apply:

$H_0[Alice/x]$ and $H_0[Bob/x]$ are dependent:

Dependent

$\forall y\ (Smoker(Alice) \lor Friend(Alice,y) \lor Jogger(y))$

$\forall y\ (Smoker(Bob) \lor Friend(Bob,y) \lor Jogger(y))$

Is this FO circuit language not powerful enough?

[Suciu'11]

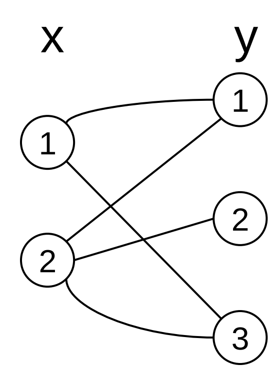
Background:
Positive Partitioned 2CNF

A PP2CNF is:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

where E = the edge set of a bipartite graph

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$



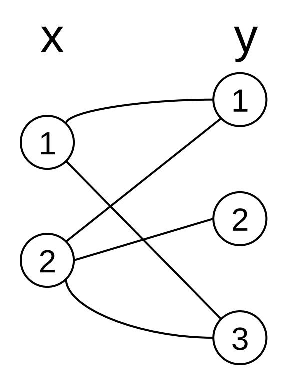
Background: Positive Partitioned 2CNF

A PP2CNF is:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

where E = the edge set of a bipartite graph

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$



Theorem: #PP2CNF is **#P**-hard [Provan'83]

Our Problematic Clause

$H_0 = \forall x \forall y \text{ Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

Our Problematic Clause

$H_0 = \forall x \forall y\ \text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

Theorem. Computing $P(H_0)$ is **#P**-hard in the size of weight function $w(.)$ (i.e., the number of people)

[Dalvi&S.'04]

Our Problematic Clause

$H_0 = \forall x \forall y\ \text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

Theorem. Computing $P(H_0)$ is **#P**-hard in the size of weight function $w(.)$ (i.e., the number of people)

[Dalvi&S.'04]

Proof: PP2CNF: $F = (X_{i1} \vee Y_{j1}) \wedge (X_{i2} \vee Y_{j2}) \wedge \ldots$ reduce #F to computing $P(H_0)$

By example:

Our Problematic Clause

$H_0 = \forall x \forall y\, \text{Smoker}(x) \lor \text{Friend}(x,y) \lor \text{Jogger}(y)$

Theorem. Computing $P(H_0)$ is **#P**-hard in the size of weight function $w(.)$ (i.e., the number of people)

[Dalvi&S.'04]

Proof: PP2CNF: $F = (X_{i1} \lor Y_{j1}) \land (X_{i2} \lor Y_{j2}) \land \ldots$ reduce #F to computing $P(H_0)$

By example:

$F = (X_1 \lor Y_1) \land (X_1 \lor Y_2) \land (X_2 \lor Y_2)$

Our Problematic Clause

$H_0 = \forall x \forall y\, \text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

Theorem. Computing $P(H_0)$ is **#P**-hard in the size of weight function w(.) (i.e., the number of people)

[Dalvi&S.'04]

Proof: PP2CNF: $F = (X_{i1} \vee Y_{j1}) \wedge (X_{i2} \vee Y_{j2}) \wedge \ldots$ reduce #F to computing $P(H_0)$

By example:

$F = (X_1 \vee Y_1) \wedge (X_1 \vee Y_2) \wedge (X_2 \vee Y_2)$

Probabilities (tuples not shown have P=1)

Smoker

X	P
x_1	0.5
x_2	0.5

Friend

X	Y	P
x_1	y_1	0
x_1	y_2	0
x_2	y_2	0

Jogger

Y	P
y_1	0.5
y_2	0.5

Our Problematic Clause

$H_0 = \forall x \forall y\, \text{Smoker}(x) \vee \text{Friend}(x,y) \vee \text{Jogger}(y)$

Theorem. Computing $P(H_0)$ is **#P**-hard in the size of weight function w(.) (i.e., the number of people)

[Dalvi&S.'04]

Proof: PP2CNF: $F = (X_{i1} \vee Y_{j1}) \wedge (X_{i2} \vee Y_{j2}) \wedge \ldots$ reduce #F to computing $P(H_0)$

By example:

$F = (X_1 \vee Y_1) \wedge (X_1 \vee Y_2) \wedge (X_2 \vee Y_2)$

$P(H_0) = P(F)$; hence $P(H_0)$ is #P-hard

Probabilities (tuples not shown have P=1)

Smoker

X	P
x_1	0.5
x_2	0.5

Friend

X	Y	P
x_1	y_1	0
x_1	y_2	0
x_2	y_2	0

Jogger

Y	P
y_1	0.5
y_2	0.5

What we know

What we know

1. Any d-D FO Circuit Q admits efficient P(Q) in the size of weight function w(.)

What we know

1. Any d-D FO Circuit Q admits efficient P(Q) in the size of weight function w(.)
2. Computing P(H_0) is **#P**-hard

What we know

1. Any d-D FO Circuit Q admits efficient P(Q) in the size of weight function w(.)
2. Computing P(H_0) is **#P**-hard
3. Therefore H_0 has no d-D FO Circuit under standard complexity assumptions

What we know

1. Any d-D FO Circuit Q admits efficient P(Q) in the size of weight function w(.)
2. Computing P(H_0) is #**P**-hard
3. Therefore H_0 has no d-D FO Circuit under standard complexity assumptions

Next: This generalizes!

Background: Hierarchical Queries

at(x) = set of atoms containing the variable x

Definition Q is **hierarchical** if for all variables x, y:

$$at(x) \subseteq at(y) \quad \text{or} \quad at(x) \supseteq at(y) \quad \text{or} \quad at(x) \cap at(y) = \emptyset$$

Background: Hierarchical Queries

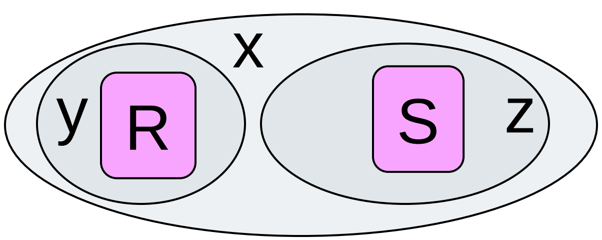
at(x) = set of atoms containing the variable x

<u>Definition</u> Q is **hierarchical** if for all variables x, y:

$at(x) \subseteq at(y)$ or $at(x) \supseteq at(y)$ or $at(x) \cap at(y) = \emptyset$

Hierarchical

$Q = \forall x\, \forall y\, \forall z\; S(x,y) \vee T(x,z)$



Background: Hierarchical Queries

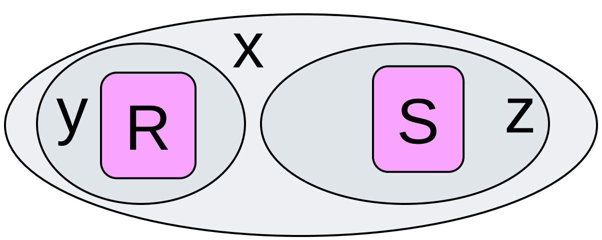
at(x) = set of atoms containing the variable x

<u>Definition</u> Q is **hierarchical** if for all variables x, y:

$$at(x) \subseteq at(y) \quad \text{or} \quad at(x) \supseteq at(y) \quad \text{or} \quad at(x) \cap at(y) = \emptyset$$

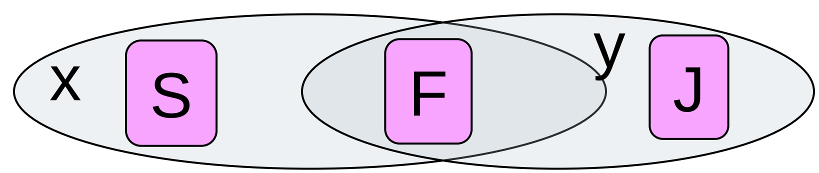
Hierarchical

$Q = \forall x\, \forall y\, \forall z\; S(x,y) \vee T(x,z)$



Non-hierarchical

$H_0 = \forall x\, \forall y\; S(x) \vee F(x,y) \vee J(y)$



The Small Dichotomy Theorem

Theorem Let Q be *one clause, with no repeated symbols*

- If Q is hierarchical, then P(Q) is in **PTIME**.
- If Q is not hierarchical then P(Q) is **#P**-hard.

[Dalvi&S.04]

The Small Dichotomy Theorem

<u>Theorem</u> Let Q be <u>*one clause, with no repeated symbols*</u>

- If Q is hierarchical, then P(Q) is in **PTIME**.
- If Q is not hierarchical then P(Q) is **#P**-hard.

[Dalvi&S.04]

<u>Corollary</u> Let Q be <u>*one clause, with no repeated symbols*</u>

- If Q is hierarchical, then Q **has a d-D FO Circuit**
- If Q is not hierarchical then Q **has no d-D FO Circuit** under standard complexity assumptions

The Small Dichotomy Theorem

Theorem Let Q be *one clause, with no repeated symbols*

- If Q is hierarchical, then P(Q) is in **PTIME**.
- If Q is not hierarchical then P(Q) is **#P**-hard.

[Dalvi&S.04]

Corollary Let Q be *one clause, with no repeated symbols*

- If Q is hierarchical, then Q **has a d-D FO Circuit**
- If Q is not hierarchical then Q **has no d-D FO Circuit** under standard complexity assumptions

Checking "Q is hierarchical" is in AC^0 (expression complexity)
Compiling the d-D FO Circuit is in PTIME

Proof

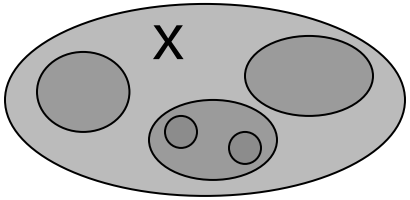
Hierarchical → **PTIME**

Proof

Hierarchical → PTIME

Case 1:

Q=



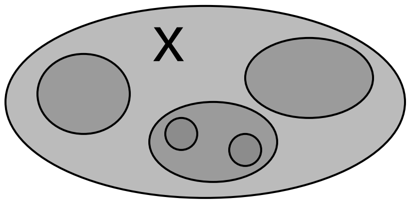
∀x must be decomposable

Proof

Hierarchical → **PTIME**

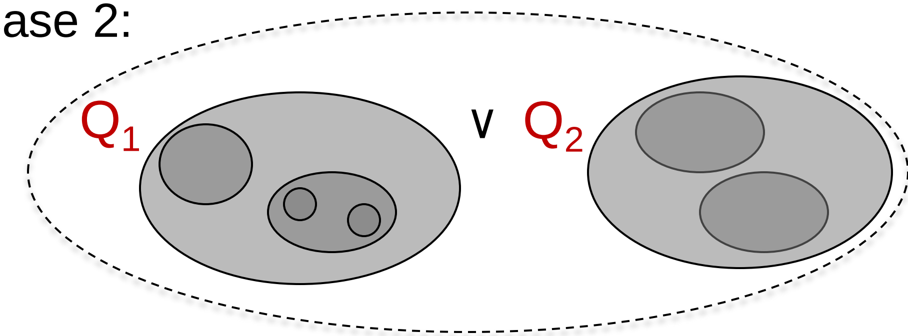
Case 1:

Q=



Case 2:

Q=



∀x must be decomposable

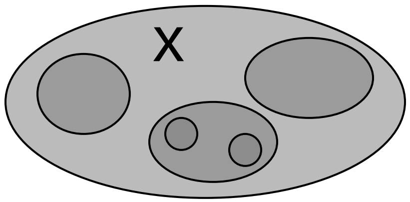
∨ must be decomposable

Proof

Hierarchical → **PTIME**

Case 1:

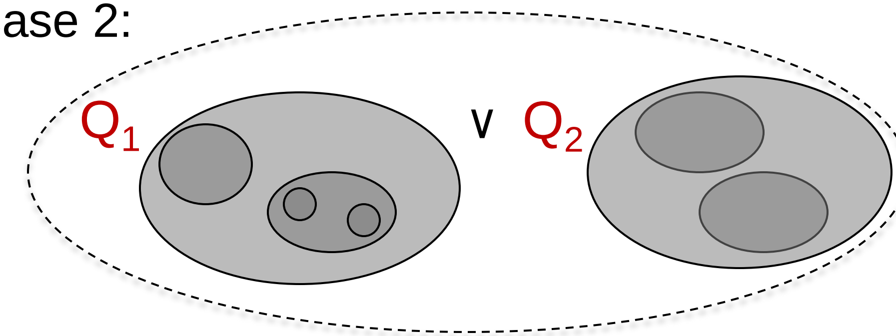
Q=



$\forall x$ must be decomposable

Case 2:

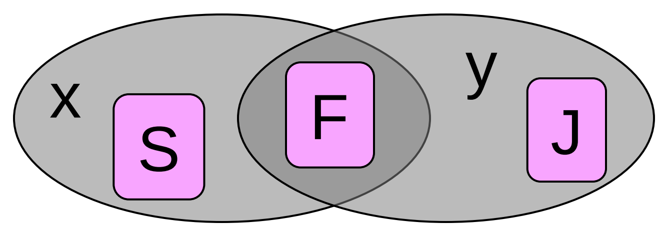
Q=



∨ must be decomposable

Non-hierarchical → **#P**-hard

Reduction from H_0:



Q = … S(x, …) ∨ F(x,y,…) ∨ J(y,…), …

Overview

1. Propositional Refresher
2. Primer: A First-Order Tractable Language
3. **Probabilistic Databases**
4. Symmetric First-Order Model Counting
5. Lots of Pointers

Probabilistic Databases

- Tuple-independent probabilistic database

Scientist

x	P
Erdos	0.9
Einstein	0.8
Pauli	0.6

Coauthor

x	y	P
Erdos	Renyi	0.6
Einstein	Pauli	0.7
Obama	Erdos	0.1

- Learned from the web, large text corpora, ontologies, etc., using **statistical** machine learning.

[Suciu'11]

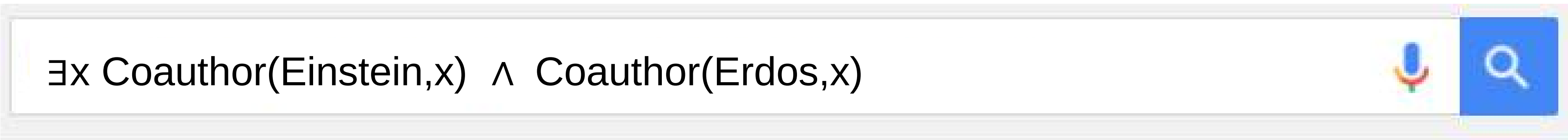
Probabilistic Databases

$\exists x\ \text{Coauthor}(\text{Einstein},x) \land \text{Coauthor}(\text{Erdos},x)$

- Conjunctive queries (CQ)

  $\exists + \land$

Probabilistic Databases



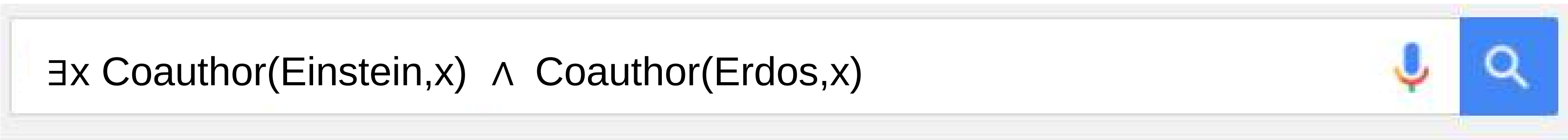
- Conjunctive queries (CQ)

  ∃ + ∧

- Unions of conjunctive queries (UCQ)

  ∨ of ∃ + ∧

Probabilistic Databases



- Conjunctive queries (CQ)

  $\exists + \wedge$

- Unions of conjunctive queries (UCQ)

  $\vee$ of $\exists + \wedge$

- Duality
  - Negation of CQ is monotone $\forall$-clause
  - Negation of UCQ is monotone $\forall$-CNF

Tuple-Independent Probabilistic DB

Probabilistic database D:

Coauthor

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Tuple-Independent Probabilistic DB

Probabilistic database D:

Coauthor

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:

x	y
A	B
A	C
B	C

$p_1p_2p_3$

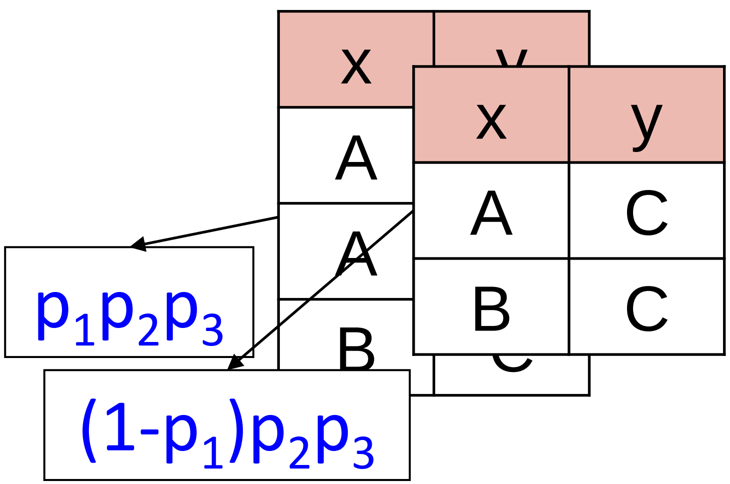
Tuple-Independent Probabilistic DB

Probabilistic database D:

Coauthor

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:



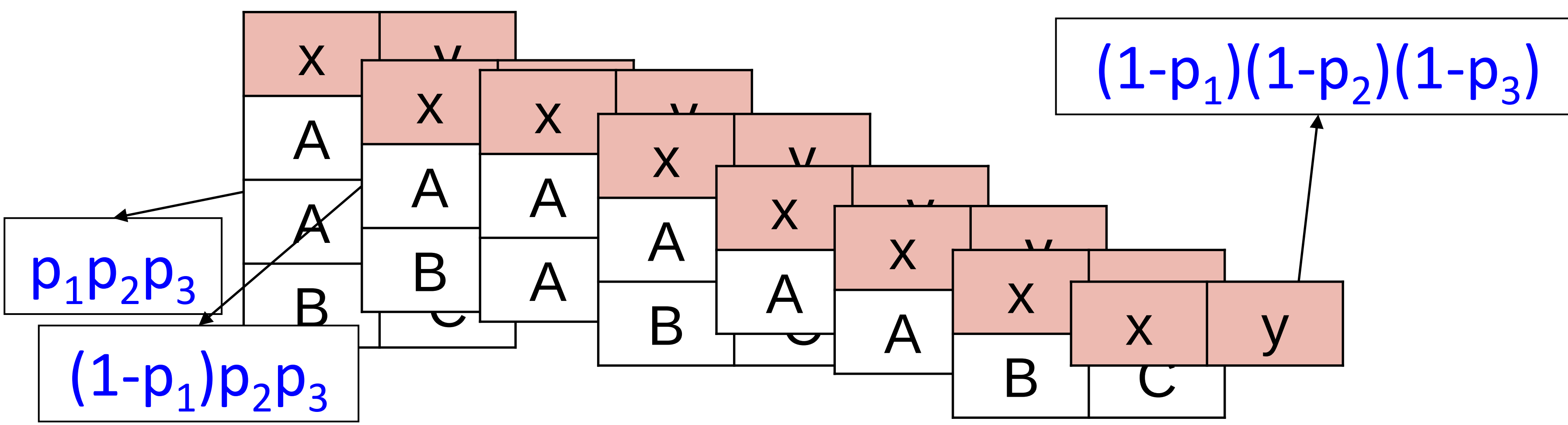
Tuple-Independent Probabilistic DB

Probabilistic database D:

Coauthor

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3

Possible worlds semantics:



Probabilistic Query Evaluation

$Q = \exists x \exists y \, Scientist(x) \wedge Coauthor(x,y)$

$P(Q) =$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Probabilistic Query Evaluation

$Q = \exists x \exists y \; Scientist(x) \wedge Coauthor(x,y)$

$P(Q) = \quad 1-(1-q_1)*(1-q_2)$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Probabilistic Query Evaluation

Q = ∃x∃y Scientist(x) ∧ Coauthor(x,y)

$$P(Q) = p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Probabilistic Query Evaluation

$Q = \exists x \exists y \; Scientist(x) \wedge Coauthor(x,y)$

$$P(Q) = p_1 * [\; 1-(1-q_1)*(1-q_2) \;]$$

$$1-(1-q_3)*(1-q_4)*(1-q_5)$$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Probabilistic Query Evaluation

$Q = \exists x \exists y \; Scientist(x) \land Coauthor(x,y)$

$$P(Q) = \quad p_1 * [\; 1-(1-q_1)*(1-q_2) \;]$$
$$p_2 * [\; 1-(1-q_3)*(1-q_4)*(1-q_5) \;]$$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Probabilistic Query Evaluation

$$Q = \exists x \exists y\ \text{Scientist}(x) \wedge \text{Coauthor}(x,y)$$

$$P(Q) = 1 - \{1 - p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]\} * \{1 - p_2 * [\ 1-(1-q_3)*(1-q_4)*(1-q_5)\ ]\}$$

Scientist

x	P
A	p_1
B	p_2
C	p_3

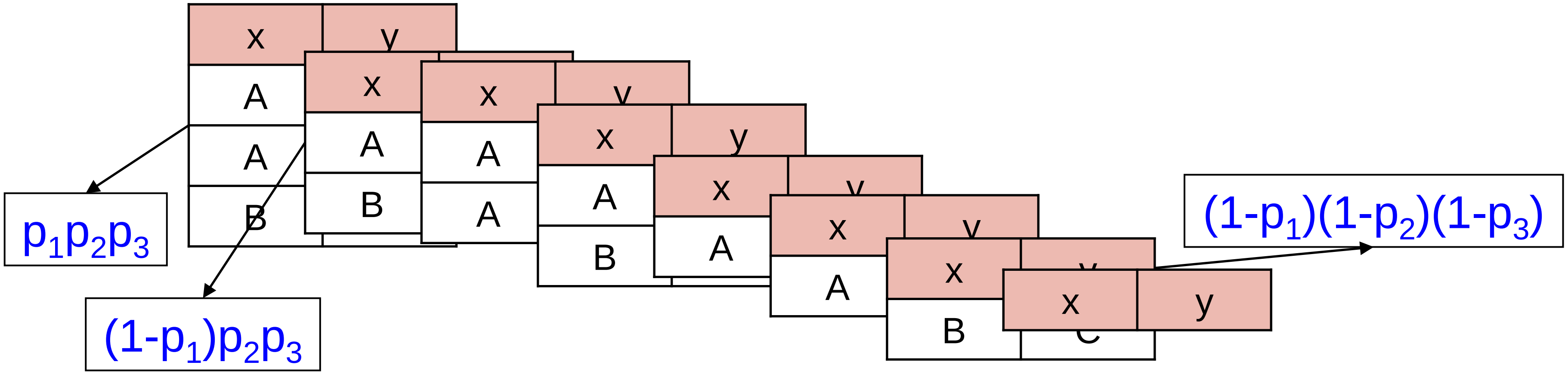
Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

From Probabilities to WMC

Friend

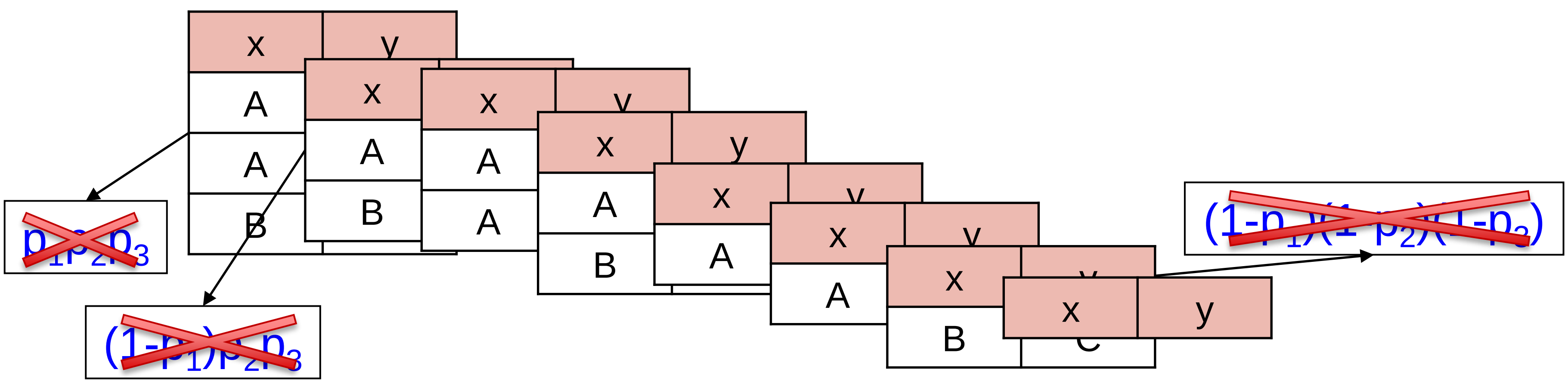
x	y	P
A	B	p_1
A	C	p_2
B	C	p_3



From Probabilities to WMC

Friend

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3



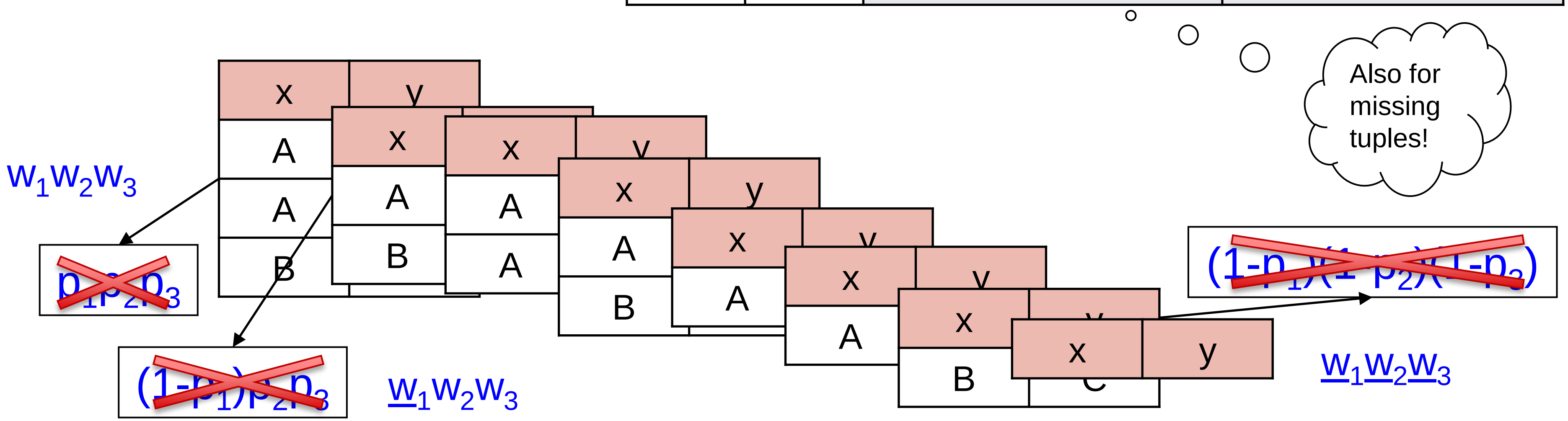
From Probabilities to WMC

Friend

x	y	P
A	B	~~p_1~~
A	C	~~p_2~~
B	C	~~p_3~~

→

x	y	w(Friend(x,y))	w(¬Friend(x,y))
A	B	$w_1 = p_1$	$\underline{w}_1 = 1\text{-}p_1$
A	C	$w_2 = p_2$	$\underline{w}_2 = 1\text{-}p_2$
B	C	$w_3 = p_3$	$\underline{w}_3 = 1\text{-}p_3$
A	A	$w_4 = 0$	$\underline{w}_4 = 1$
A	C	$w_5 = 0$	$\underline{w}_5 = 1$
	…	…	



Lifted Inference Rules

Preprocess Q (omitted)

[Suciu'11]

Lifted Inference Rules

Preprocess Q (omitted)

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$	Negation
$P(Q1 \wedge Q2) = P(Q1)\, P(Q2)$ $P(Q1 \vee Q2) = 1 - (1 - P(Q1))\,(1 - P(Q2))$	Decomposable ∧,∨
$P(\forall z\, Q) = \Pi_{A \in Domain}\, P(Q[A/z])$ $P(\exists z\, Q) = 1 - \Pi_{A \in Domain}\, (1 - P(Q[A/z]))$	Decomposable ∀,∃
$P(Q1 \wedge Q2) = P(Q1) + P(Q2) - 1$ $P(Q1 \vee Q2) = P(Q1) + P(Q2)$	Deterministic ∧,∨
$P(\forall z\, Q) = 1 - \sum_{A \in Domain} 1 - P(Q[A/z])$ $P(\exists z\, Q) = \sum_{A \in Domain} P(Q[A/z])$	Deterministic ∀,∃

Decomposability

Determinism

[Suciu'11]

Lifted Inference Rules

Preprocess Q (omitted)

Evaluate Probability on FO Circuit*

$P(\neg Q) = 1 - P(Q)$	Negation
$P(Q1 \wedge Q2) = P(Q1)\, P(Q2)$ $P(Q1 \vee Q2) = 1 - (1 - P(Q1))\,(1 - P(Q2))$	Decomposable ∧,∨
$P(\forall z\, Q) = \Pi_{A \in Domain}\ P(Q[A/z])$ $P(\exists z\, Q) = 1 - \Pi_{A \in Domain}\ (1 - P(Q[A/z]))$	Decomposable ∀,∃
$P(Q1 \wedge Q2) = P(Q1) + P(Q2) - 1$ $P(Q1 \vee Q2) = P(Q1) + P(Q2)$	Deterministic ∧,∨
$P(\forall z\, Q) = 1 - \sum_{A \in Domain}\ 1\text{-}P(Q[A/z])$ $P(\exists z\, Q) = \sum_{A \in Domain} P(Q[A/z])$	Deterministic ∀,∃

Decomposability

Determinism

$$P(Q1 \wedge Q2) = P(Q1) + P(Q2) - P(Q1 \vee Q2)$$
$$P(Q1 \vee Q2) = P(Q1) + P(Q2) - P(Q1 \wedge Q2)$$

Inclusion/ Exclusion

[Suciu'11]

Lifted Inference Rules

Preprocess Q (omitted)

Evaluate Probability on FO Circuit*

Rule	
$P(\neg Q) = 1 - P(Q)$	Negation
$P(Q1 \land Q2) = P(Q1)\,P(Q2)$ $P(Q1 \lor Q2) = 1 - (1 - P(Q1))\,(1 - P(Q2))$	Decomposable ∧,∨
$P(\forall z\, Q) = \Pi_{A \in Domain}\, P(Q[A/z])$ $P(\exists z\, Q) = 1 - \Pi_{A \in Domain}\,(1 - P(Q[A/z]))$	Decomposable ∀,∃
$P(Q1 \land Q2) = P(Q1) + P(Q2) - 1$ $P(Q1 \lor Q2) = P(Q1) + P(Q2)$	Deterministic ∧,∨
$P(\forall z\, Q) = 1 - \sum_{A \in Domain} 1 - P(Q[A/z])$ $P(\exists z\, Q) = \sum_{A \in Domain} P(Q[A/z])$	Deterministic ∀,∃

Decomposability

Determinism

$$P(Q1 \land Q2) = P(Q1) + P(Q2) - P(Q1 \lor Q2)$$
$$P(Q1 \lor Q2) = P(Q1) + P(Q2) - P(Q1 \land Q2)$$

Inclusion/
Exclusion

[Suciu'11]

Background: **#P**-hard Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

Will drop ∀ to reduce clutter

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

Background: #**P**-hard Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

Will drop ∀ to reduce clutter

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

$H_2 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \vee [S_2(x_2,y_2) \vee T(y_2)]$

Background: **#P**-hard Queries H_k

Will drop $\forall$ to reduce clutter

$H_0 = R(x) \vee S(x,y) \vee T(y)$

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

$H_2 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \vee [S_2(x_2,y_2) \vee T(y_2)]$

$$\begin{aligned} H_3 = {} & [R(x_0) \vee S_1(x_0,y_0)] \\ & \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \\ & \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \\ & \wedge [S_3(x_3,y_3) \vee T(y_3)] \end{aligned}$$

. . .

Background: #**P**-hard Queries H_k

$H_0 = R(x) \lor S(x,y) \lor T(y)$

Will drop ∀ to reduce clutter

$H_1 = [R(x_0) \lor S(x_0,y_0)] \land [S(x_1,y_1) \lor T(y_1)]$

$H_2 = [R(x_0) \lor S_1(x_0,y_0)] \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \lor [S_2(x_2,y_2) \lor T(y_2)]$

$$H_3 = [R(x_0) \lor S_1(x_0,y_0)] \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \land [S_2(x_2,y_2) \lor S_3(x_2,y_2)] \land [S_3(x_3,y_3) \lor T(y_3)]$$

. . .

Theorem. Every query H_k is #**P**-hard

[Dalvi&S'12]

I/E and Cancellations

$$
\begin{aligned}
Q_W = {} & [(R(x_0) \lor S_1(x_0,y_0)) \land (S_2(x_2,y_2) \lor S_3(x_2,y_2))] && Q_1 \\
& \lor [(R(x_0) \lor S_1(x_0,y_0)) \land (S_3(x_3,y_3) \lor T(y_3))] && Q_2 \\
& \lor [(S_1(x_1,y_1) \lor S_2(x_1,y_1)) \land (S_3(x_3,y_3) \lor T(y_3))] && Q_3
\end{aligned}
$$

[Suciu'11]

I/E and Cancellations

$$Q_W = [(R(x_0) \lor S_1(x_0,y_0)) \land (S_2(x_2,y_2) \lor S_3(x_2,y_2))] \quad Q_1$$
$$\lor [(R(x_0) \lor S_1(x_0,y_0)) \land (S_3(x_3,y_3) \lor T(y_3))] \quad Q_2$$
$$\lor [(S_1(x_1,y_1) \lor S_2(x_1,y_1)) \land (S_3(x_3,y_3) \lor T(y_3))] \quad Q_3$$

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) + - P(Q_1 \land Q_2) - P(Q_2 \land Q_3) - P(Q_1 \land Q_3) + P(Q_1 \land Q_2 \land Q_3)$$

[Suciu'11]

I/E and Cancellations

$$Q_W = [(R(x_0) \lor S_1(x_0,y_0)) \land (S_2(x_2,y_2) \lor S_3(x_2,y_2))] \quad Q_1$$
$$\lor [(R(x_0) \lor S_1(x_0,y_0)) \land (S_3(x_3,y_3) \lor T(y_3))] \quad Q_2$$
$$\lor [(S_1(x_1,y_1) \lor S_2(x_1,y_1)) \land (S_3(x_3,y_3) \lor T(y_3))] \quad Q_3$$

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) +$$
$$- P(Q_1 \land Q_2) - P(Q_2 \land Q_3) - P(Q_1 \land Q_3)$$
$$+ P(Q_1 \land Q_2 \land Q_3)$$

$= H_3$ (#**P**-hard !)

[Suciu'11]

I/E and Cancellations

$$
\begin{aligned}
Q_W = & [(R(x_0) \vee S_1(x_0,y_0)) \wedge (S_2(x_2,y_2) \vee S_3(x_2,y_2))] \quad Q_1\\
& \vee [(R(x_0) \vee S_1(x_0,y_0)) \wedge (S_3(x_3,y_3) \vee T(y_3))] \quad Q_2\\
& \vee [(S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge (S_3(x_3,y_3) \vee T(y_3))] \quad Q_3
\end{aligned}
$$

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \wedge Q_2) - P(Q_2 \wedge Q_3) - \cancel{P(Q_1 \wedge Q_3)} \\
& \cancel{+ P(Q_1 \wedge Q_2 \wedge Q_3)}
\end{aligned}
$$

$= H_3$ (#**P**-hard !)

Also $= H_3$

[Suciu'11]

I/E and Cancellations

$$Q_W = [(R(x_0) \vee S_1(x_0,y_0)) \wedge (S_2(x_2,y_2) \vee S_3(x_2,y_2))] \quad Q_1$$
$$\vee [(R(x_0) \vee S_1(x_0,y_0)) \wedge (S_3(x_3,y_3) \vee T(y_3))] \quad Q_2$$
$$\vee [(S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge (S_3(x_3,y_3) \vee T(y_3))] \quad Q_3$$

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) +$$
$$- P(Q_1 \wedge Q_2) - P(Q_2 \wedge Q_3) - \cancel{P(Q_1 \wedge Q_3)}$$
$$\cancel{+ P(Q_1 \wedge Q_2 \wedge Q_3)}$$

Also = H_3

= H_3 (#**P**-hard !)

Need to cancel terms to compute the query in **PTIME**

Using Mobius' function on the implication lattice of Q_W

[Suciu'11]

The Big Dichotomy Theorem

Call Q *liftable* if the rules don't get stuck.

Dichotomy Theorem Fix a UCQ query Q.

1. If Q is **liftable**, then P(Q) is in **PTIME**
2. If Q is **not liftable**, then P(Q) is **#P**-complete

[Dalvi'12]

The Big Dichotomy Theorem

Call Q *liftable* if the rules don't get stuck.

Dichotomy Theorem Fix a UCQ query Q.

1. If Q is **liftable**, then P(Q) is in **PTIME**
2. If Q is **not liftable**, then P(Q) is **#P**-complete

[Dalvi'12]

Lifted inference rules are complete for UCQ!

Open Problem

- For CQs w/o repeated symbols, **PTIME** Q = FO circuit language
- We need inclusion/exclusion to capture **PTIME** UCQs
- I/E is arithmetic operation

P(Q1) + P(Q2) - P(Q1 v Q2)

Open Problem

- For CQs w/o repeated symbols, **PTIME** Q = FO circuit language
- We need inclusion/exclusion to capture **PTIME** UCQs
- I/E is arithmetic operation

P(Q1) + P(Q2) - P(Q1 v Q2)

What is the logical equivalent of inclusion-exclusion?
What is the circuit language capturing **PTIME** UCQs?

Open Problem

- For CQs w/o repeated symbols, **PTIME** Q = FO circuit language
- We need inclusion/exclusion to capture **PTIME** UCQs
- I/E is arithmetic operation

P(Q1) + P(Q2) - P(Q1 v Q2)

What is the logical equivalent of inclusion-exclusion?
What is the circuit language capturing **PTIME** UCQs?

- It is not decision-DNNF! (see Beame)

Linear Data Complexity

$Q = \exists x\, \exists y\, \text{Scientist}(x) \land \text{Coauthor}(x,y)$

$$P(Q) = 1 - \Pi_{A \in \text{Domain}} (1 - P(\text{Scientist}(A) \land \exists y\, \text{Coauthor}(A,y))$$

$$\begin{aligned}
= 1 - &(1 - P(\text{Scientist}(A) \land \exists y\, \text{Coauthor}(A,y))\\
&\times (1 - P(\text{Scientist}(B) \land \exists y\, \text{Coauthor}(B,y))\\
&\times (1 - P(\text{Scientist}(C) \land \exists y\, \text{Coauthor}(C,y))\\
&\times (1 - P(\text{Scientist}(D) \land \exists y\, \text{Coauthor}(D,y))\\
&\times (1 - P(\text{Scientist}(E) \land \exists y\, \text{Coauthor}(E,y))\\
&\times (1 - P(\text{Scientist}(F) \land \exists y\, \text{Coauthor}(F,y))\\
&\quad \ldots
\end{aligned}$$

[Ceylan’16]

Linear Data Complexity

$$Q = \exists x\, \exists y\; Scientist(x) \wedge Coauthor(x,y)$$

$$P(Q) = 1 - \Pi_{A \in Domain}\, (1 - P(Scientist(A) \wedge \exists y\, Coauthor(A,y))$$

$$= 1 - (1 - P(Scientist(A) \wedge \exists y\, Coauthor(A,y))$$
$$\times (1 - P(Scientist(B) \wedge \exists y\, Coauthor(B,y))$$
$$\times (1 - P(Scientist(C) \wedge \exists y\, Coauthor(C,y))$$
$$\times (1 - P(Scientist(D) \wedge \exists y\, Coauthor(D,y))$$
$$\times (1 - P(Scientist(E) \wedge \exists y\, Coauthor(E,y))$$
$$\times (1 - P(Scientist(F) \wedge \exists y\, Coauthor(F,y))$$
$$\ldots$$

⇨ No supporting facts in database!

[Ceylan'16]

Linear Data Complexity

$Q = \exists x \, \exists y \, \text{Scientist}(x) \wedge \text{Coauthor}(x,y)$

$$P(Q) = 1 - \Pi_{A \in \text{Domain}} (1 - P(\text{Scientist}(A) \wedge \exists y \, \text{Coauthor}(A,y))$$

$= 1 - (1 - P(\text{Scientist}(A) \wedge \exists y \, \text{Coauthor}(A,y))$

$\times (1 - P(\text{Scientist}(B) \wedge \exists y \, \text{Coauthor}(B,y))$

$\times (1 - P(\text{Scientist}(C) \wedge \exists y \, \text{Coauthor}(C,y))$

$\times (1 - P(\text{Scientist}(D) \wedge \exists y \, \text{Coauthor}(D,y))$

$\times (1 - P(\text{Scientist}(E) \wedge \exists y \, \text{Coauthor}(E,y))$

$\times (1 - P(\text{Scientist}(F) \wedge \exists y \, \text{Coauthor}(F,y))$

…

⇨ No supporting facts in database!

⇨ Probability 0

[Ceylan'16]

Linear Data Complexity

$Q = \exists x\ \exists y\ Scientist(x) \land Coauthor(x,y)$

$P(Q) = 1 - \prod_{A \in Domain} (1 - P(Scientist(A) \land \exists y\ Coauthor(A,y))$

$= 1 - (1 - P(Scientist(A) \land \exists y\ Coauthor(A,y))$
$\times (1 - P(Scientist(B) \land \exists y\ Coauthor(B,y))$
$\times (1 - P(Scientist(C) \land \exists y\ Coauthor(C,y))$
$\times (1 - P(Scientist(D) \land \exists y\ Coauthor(D,y))$
$\times (1 - P(Scientist(E) \land \exists y\ Coauthor(E,y))$
$\times (1 - P(Scientist(F) \land \exists y\ Coauthor(F,y))$
…

⇒ No supporting facts in database!

⇒ Probability 0

⇒ Ignore these sub-queries!

[Ceylan'16]

Linear Data Complexity

$Q = \exists x\, \exists y\, \mathrm{Scientist}(x) \wedge \mathrm{Coauthor}(x,y)$

$$P(Q) = 1 - \Pi_{A \in \mathrm{Domain}} (1 - P(\mathrm{Scientist}(A) \wedge \exists y\, \mathrm{Coauthor}(A,y))$$

$= 1 - (1 - P(\mathrm{Scientist}(A) \wedge \exists y\, \mathrm{Coauthor}(A,y))$
$\times (1 - P(\mathrm{Scientist}(B) \wedge \exists y\, \mathrm{Coauthor}(B,y))$
$\times (1 - P(\mathrm{Scientist}(C) \wedge \exists y\, \mathrm{Coauthor}(C,y))$
$\times (1 - P(\mathrm{Scientist}(D) \wedge \exists y\, \mathrm{Coauthor}(D,y))$
$\times (1 - P(\mathrm{Scientist}(E) \wedge \exists y\, \mathrm{Coauthor}(E,y))$
$\times (1 - P(\mathrm{Scientist}(F) \wedge \exists y\, \mathrm{Coauthor}(F,y))$
…

⇒ No supporting facts in database!

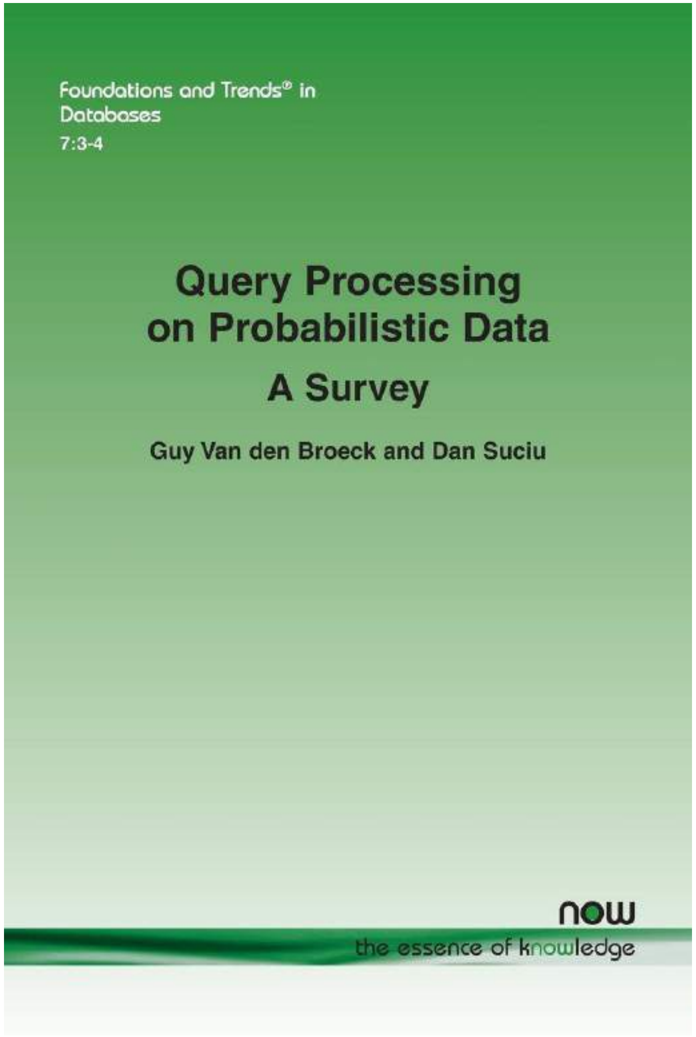
⇒ Probability 0

⇒ Ignore these sub-queries!

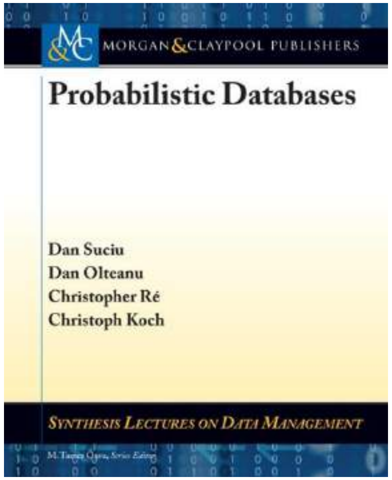
Complexity linear time in database size!

[Ceylan'16]

Commercial Break



- Survey book (2017)
- Survey book (2011)

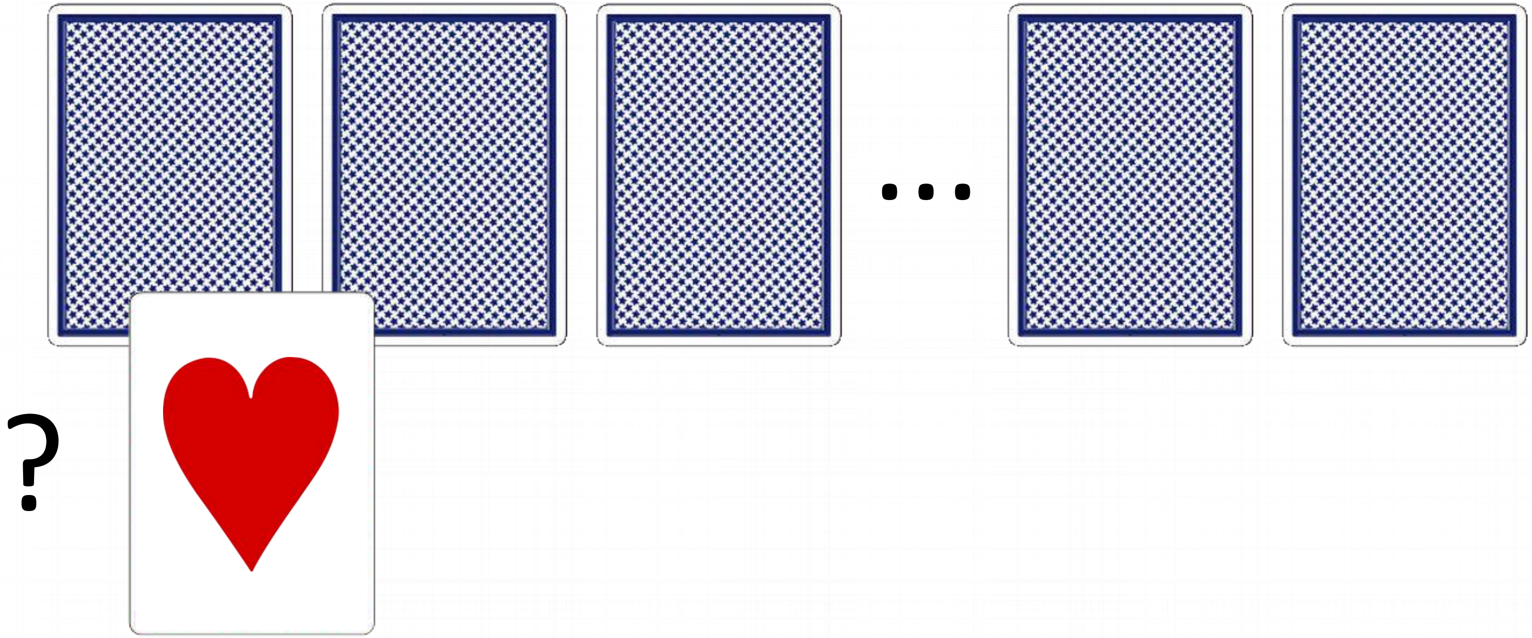


- IJCAI 2016 tutorial
  http://web.cs.ucla.edu/~guyvdb/talks/IJCAI16-tutorial/

Overview

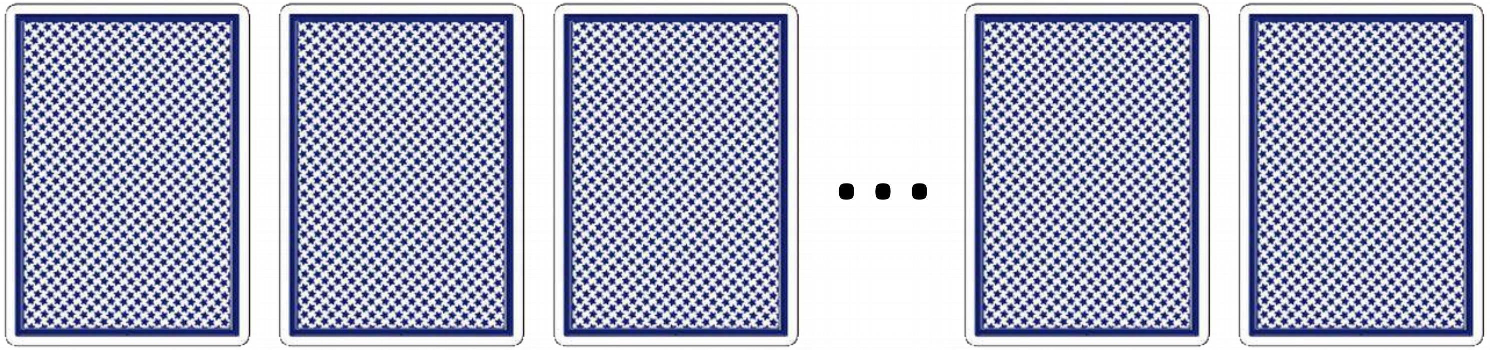
1. Propositional Refresher
2. Primer: A First-Order Tractable Language
3. Probabilistic Databases
4. **Symmetric First-Order Model Counting**
5. Lots of Pointers

Simple Reasoning Problem



Probability that Card1 is Hearts? 1/4

[Van den Broeck; AAAI-KRR'15]



Model distribution by FOMC:

$$\Delta = \forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \land Card(p,c') \Rightarrow c = c'$$

[Van den Broeck 2015]

Beyond NP Pipeline for #P

Reduce to propositional model counting:

[Van den Broeck 2015]

Beyond NP Pipeline for #P

Reduce to propositional model counting:

$$
\begin{aligned}
\Delta = \;& \text{Card}(A♥,p_1) \lor \dots \lor \text{Card}(2♣,p_1) \\
& \text{Card}(A♥,p_2) \lor \dots \lor \text{Card}(2♣,p_2) \\
& \dots \\
& \text{Card}(A♥,p_1) \lor \dots \lor \text{Card}(A♥,p_{52}) \\
& \text{Card}(K♥,p_1) \lor \dots \lor \text{Card}(K♥,p_{52}) \\
& \dots \\
& \neg\text{Card}(A♥,p_1) \lor \neg\text{Card}(A♥,p_2) \\
& \neg\text{Card}(A♥,p_1) \lor \neg\text{Card}(A♥,p_3) \\
& \dots
\end{aligned}
$$

[Van den Broeck 2015]

Beyond NP Pipeline for #P

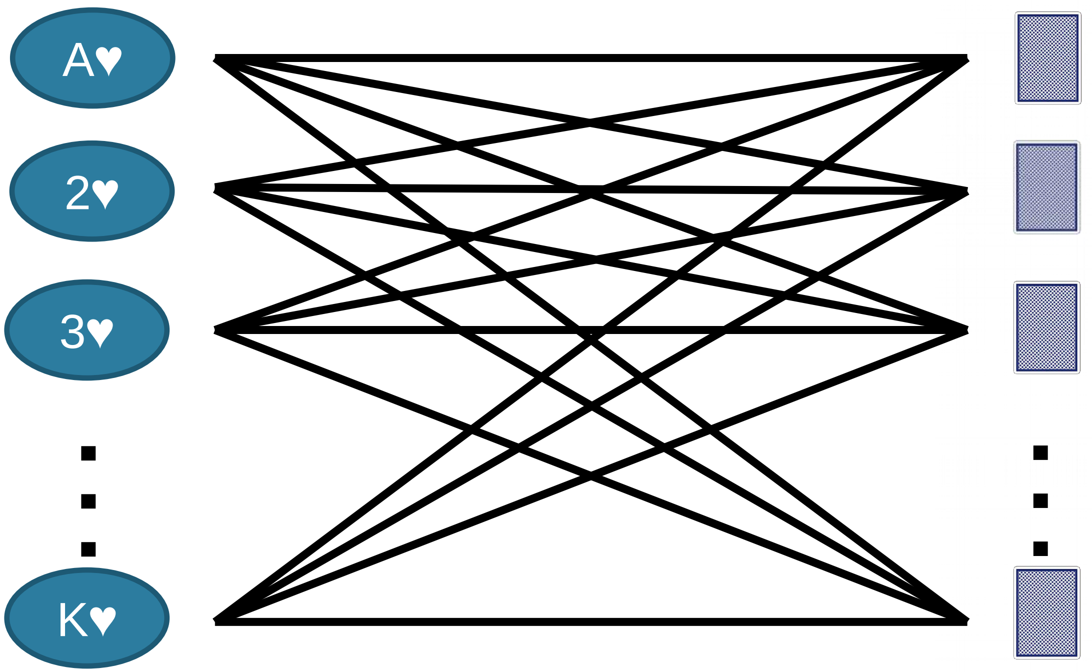
Reduce to propositional model counting:

Δ = Card(A♥,p_1) v … v Card(2♣,p_1)
Card(A♥,p_2) v … v Card(2♣,p_2)
…
Card(A♥,p_1) v … v Card(A♥,p_{52})
Card(K♥,p_1) v … v Card(K♥,p_{52})
…
¬Card(A♥,p_1) v ¬Card(A♥,p_2)
¬Card(A♥,p_1) v ¬Card(A♥,p_3)
…

What will happen?

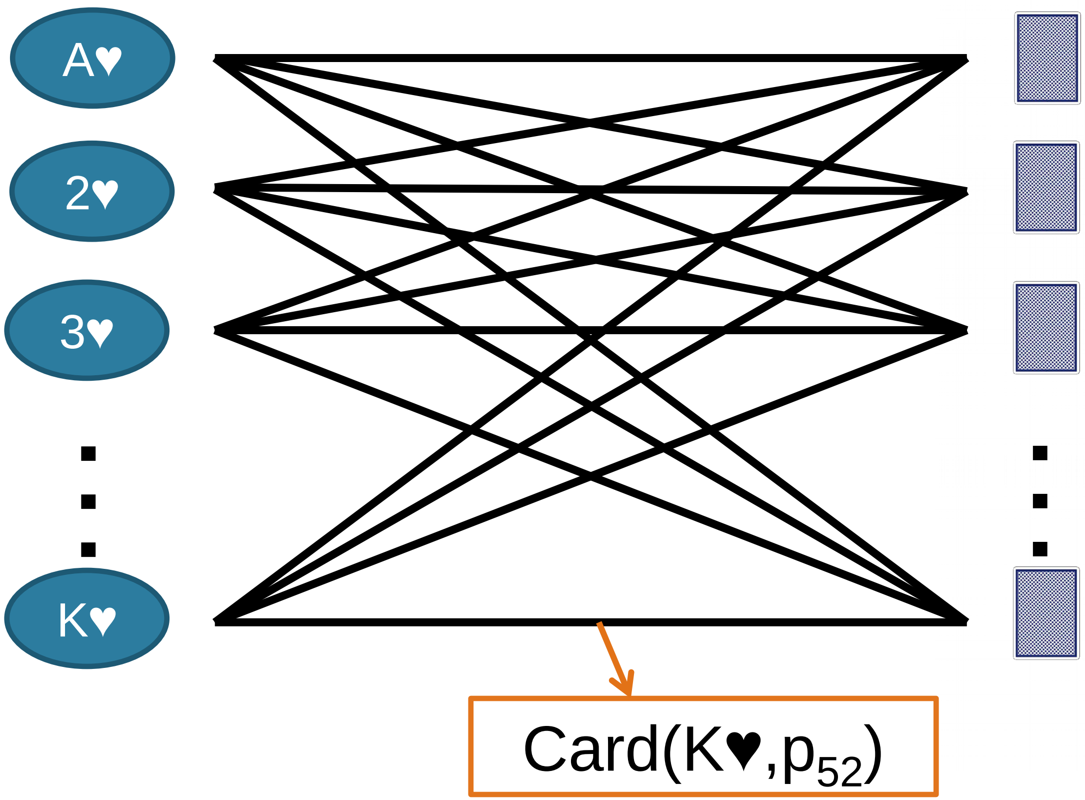
[Van den Broeck 2015]

Deck of Cards Graphically



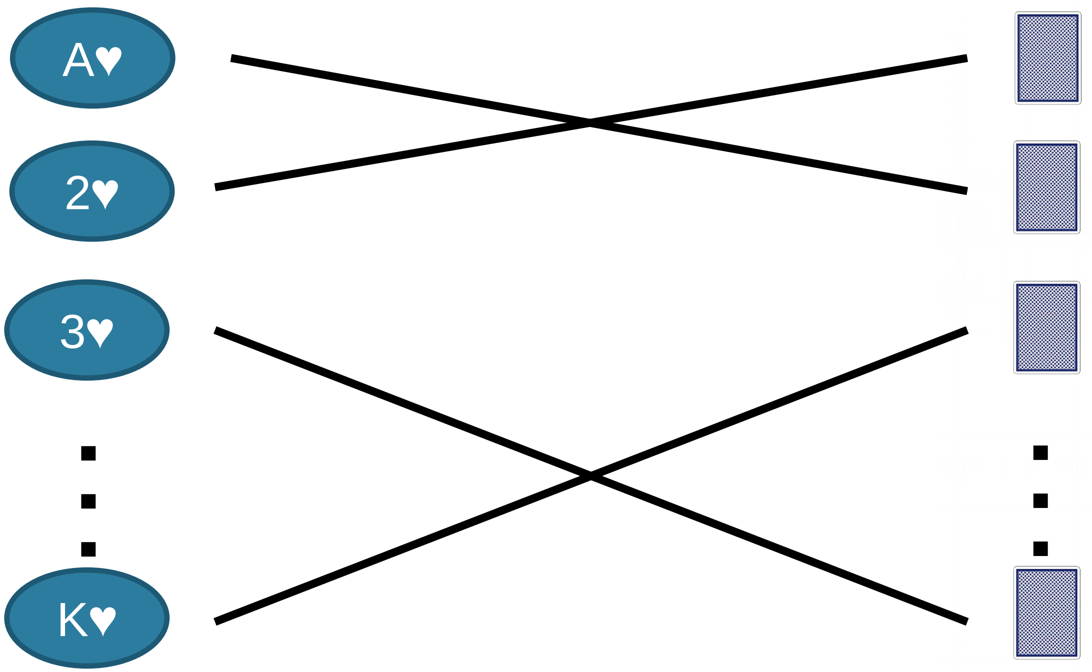
[VdB'15]

Deck of Cards Graphically



[VdB'15]

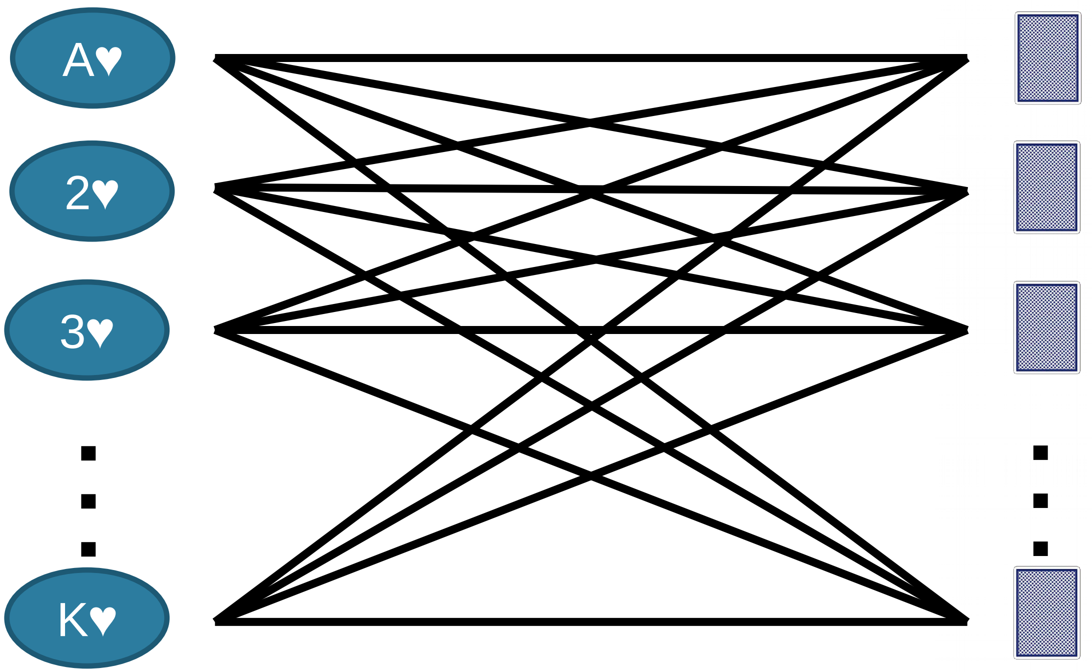
Deck of Cards Graphically



One model/*perfect matching*

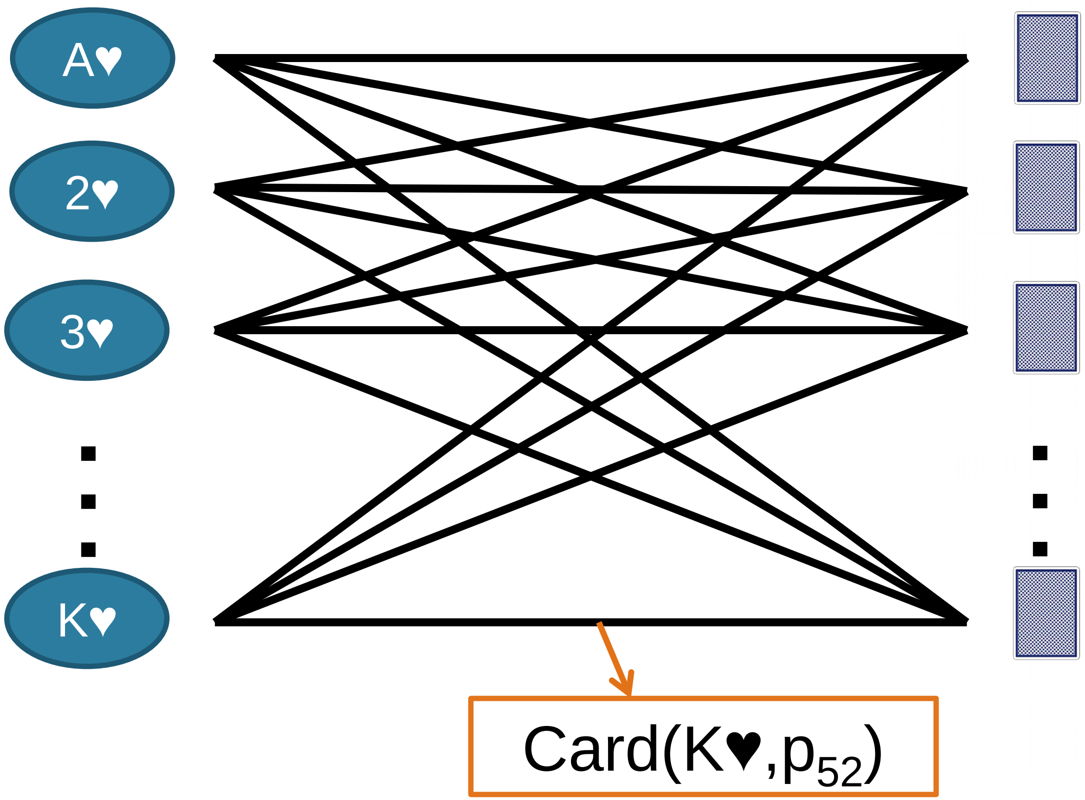
[VdB'15]

Deck of Cards Graphically



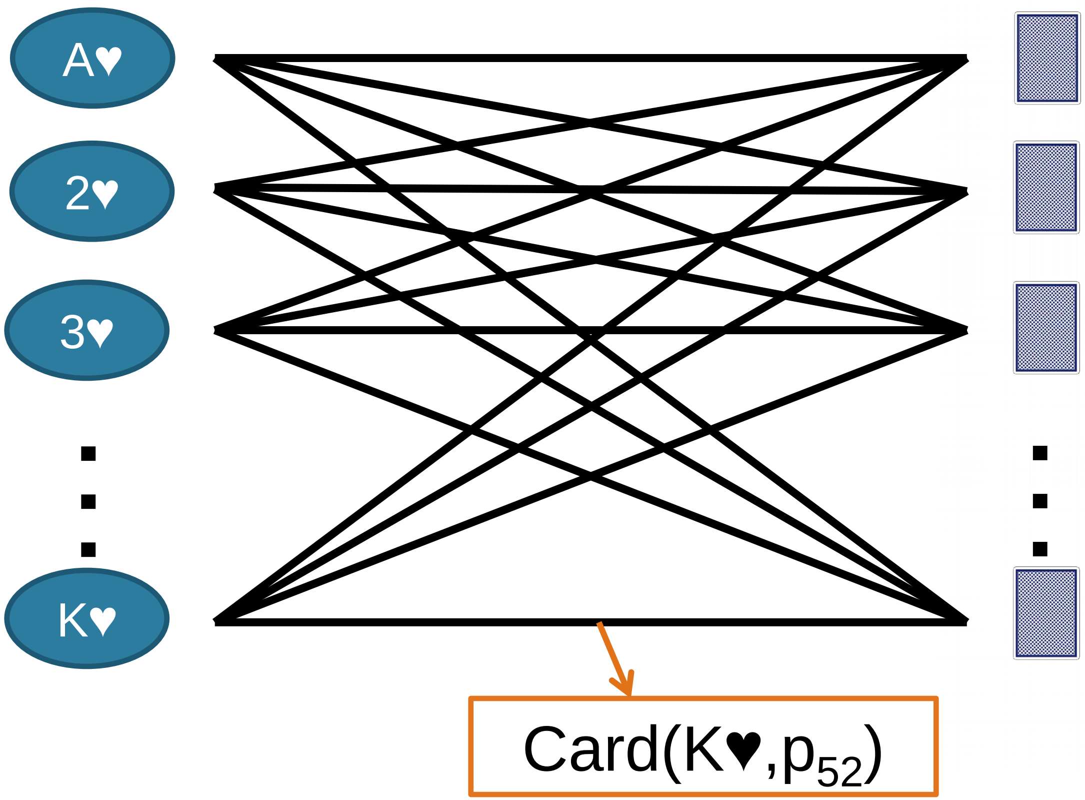
[VdB'15]

Deck of Cards Graphically



[VdB'15]

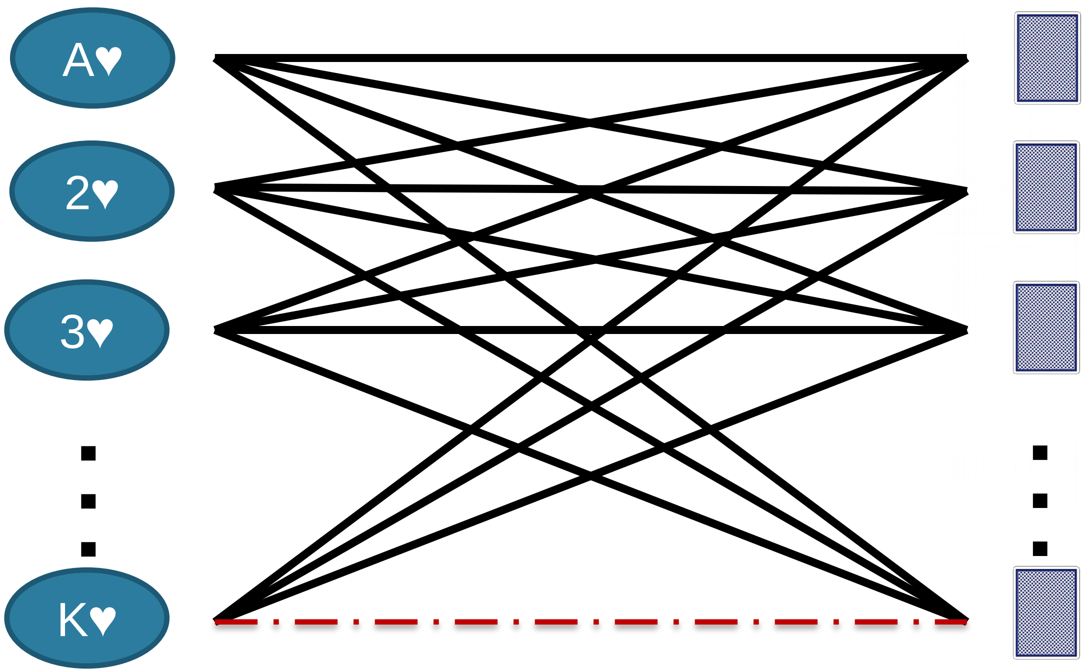
Deck of Cards Graphically



Model counting: How many *perfect matchings*?

[VdB'15]

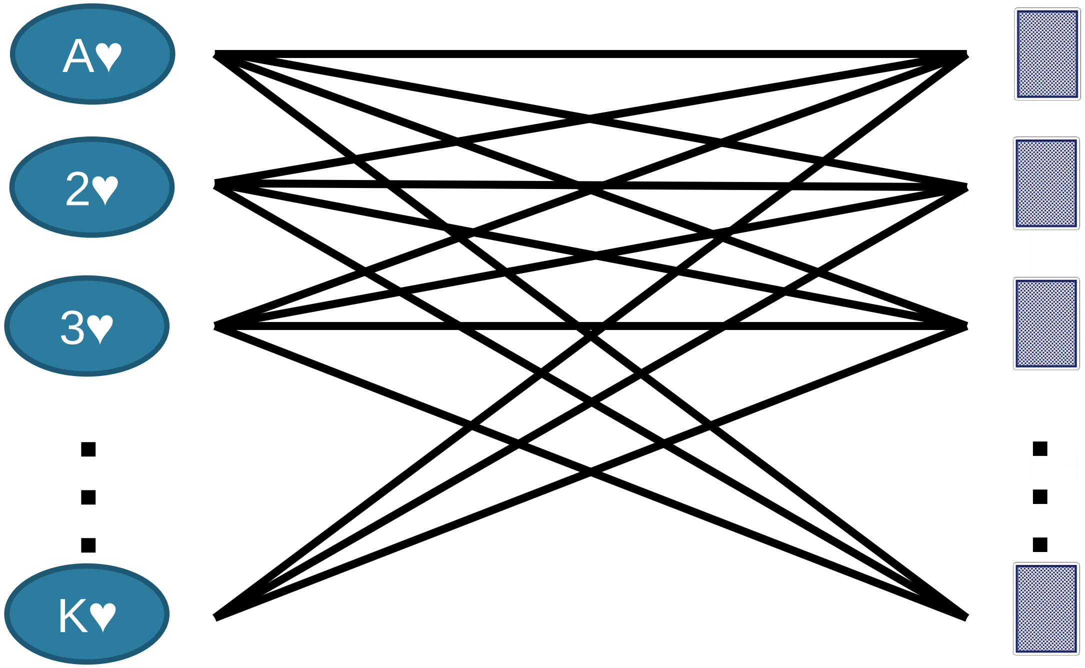
Deck of Cards Graphically



What if I set
$w(\mathrm{Card}(K♥, p_{52})) = 0$?

[VdB'15]

Deck of Cards Graphically



What if I set
$w(\text{Card}(K♥, p_{52})) = 0$?

[VdB'15]

Observations

- Weight function = bipartite graph
- # models = # perfect matchings
- Problem is **#P**-complete! ☹

[VdB'15]

Observations

- Weight function = bipartite graph
- # models = # perfect matchings
- Problem is **#P**-complete! ☹

No propositional WMC solver can handle cards problem efficiently!

[VdB'15]

Observations

- Weight function = bipartite graph
- # models = # perfect matchings
- Problem is **#P**-complete! ☹

No propositional WMC solver can handle cards problem efficiently!

What is going on here?

[VdB'15]

Symmetric Weighted FOMC

No database! No literal-specific weights!

Def. A *weighted vocabulary* is (**R**, **w**), where

- **R** = $(R_1, R_2, \ldots, R_k)$ = relational vocabulary
- **w** = $(w_1, w_2, \ldots, w_k)$ = weights
- Implicit weights: $w(R_i(t)) = w_i$

Special case: $w_i = 1$ is model counting

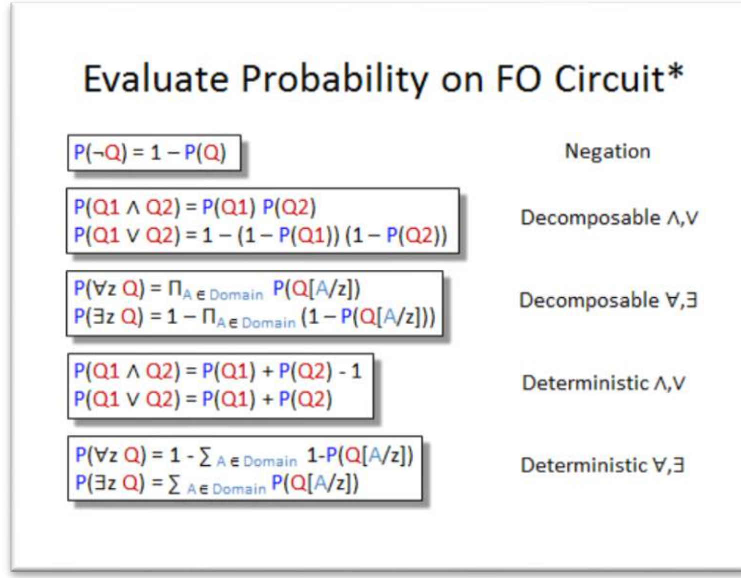
Complexity in terms of domain size n

FOMC Inference Rules

- Simplification to ∃,∀ rules:

For example:

$P(\forall z\ Q) = P(Q[C_1/z])^{|Domain|}$

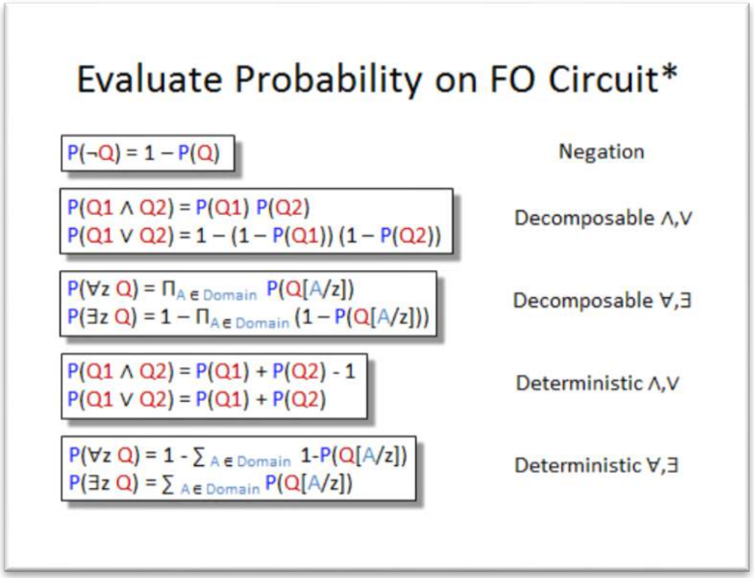


[VdB’11]

FOMC Inference Rules

- Simplification to ∃,∀ rules:

  For example:
  $P(\forall z\ Q) = P(Q[C_1/z])^{|Domain|}$



- A powerful new inference rule: *atom counting*
  Only possible with symmetric weights
  Intuition: **Remove unary relations**

The workhorse of FOMC

[VdB’11]

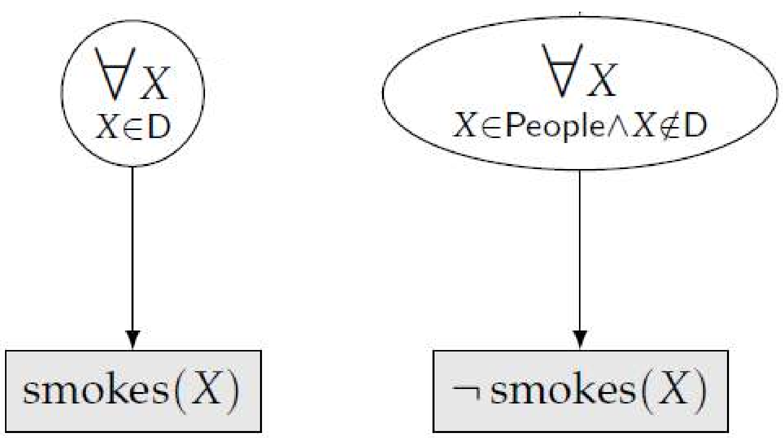
Deterministic Decomposable FO NNF

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

[Van den Broeck 2013]

Deterministic Decomposable FO NNF

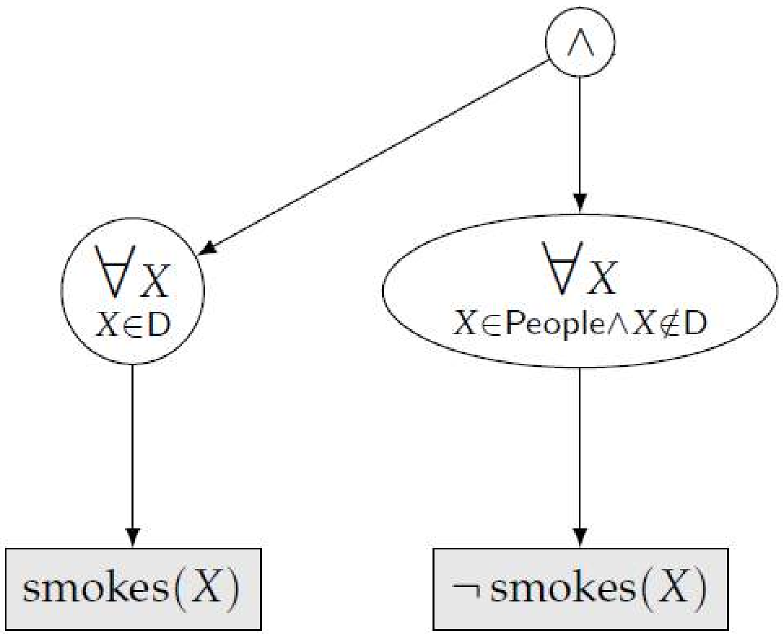
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[Van den Broeck 2013]

Deterministic Decomposable FO NNF

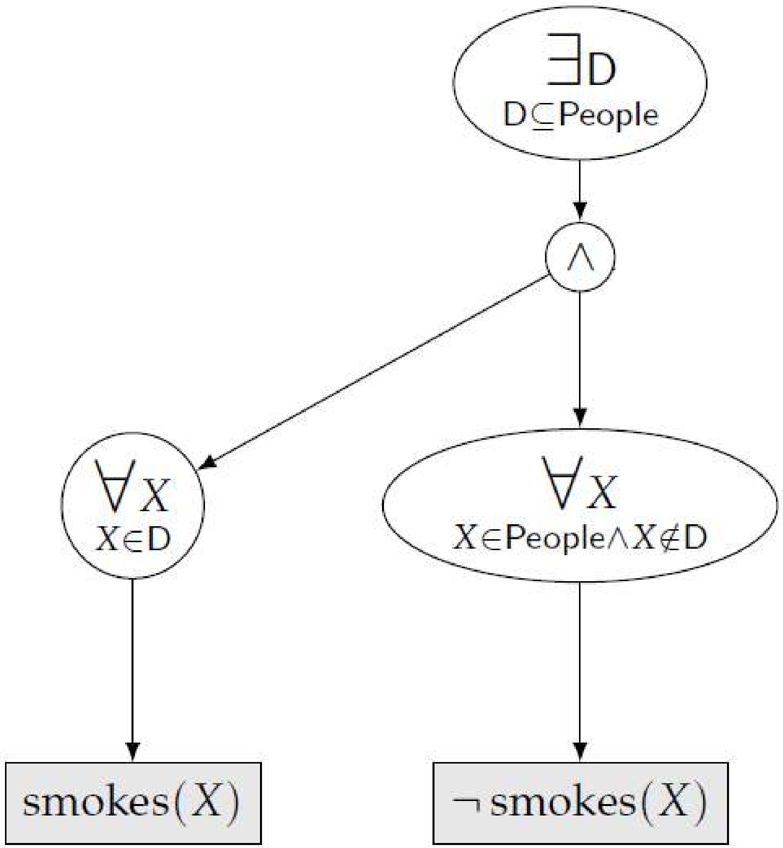
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[Van den Broeck 2013]

Deterministic Decomposable FO NNF

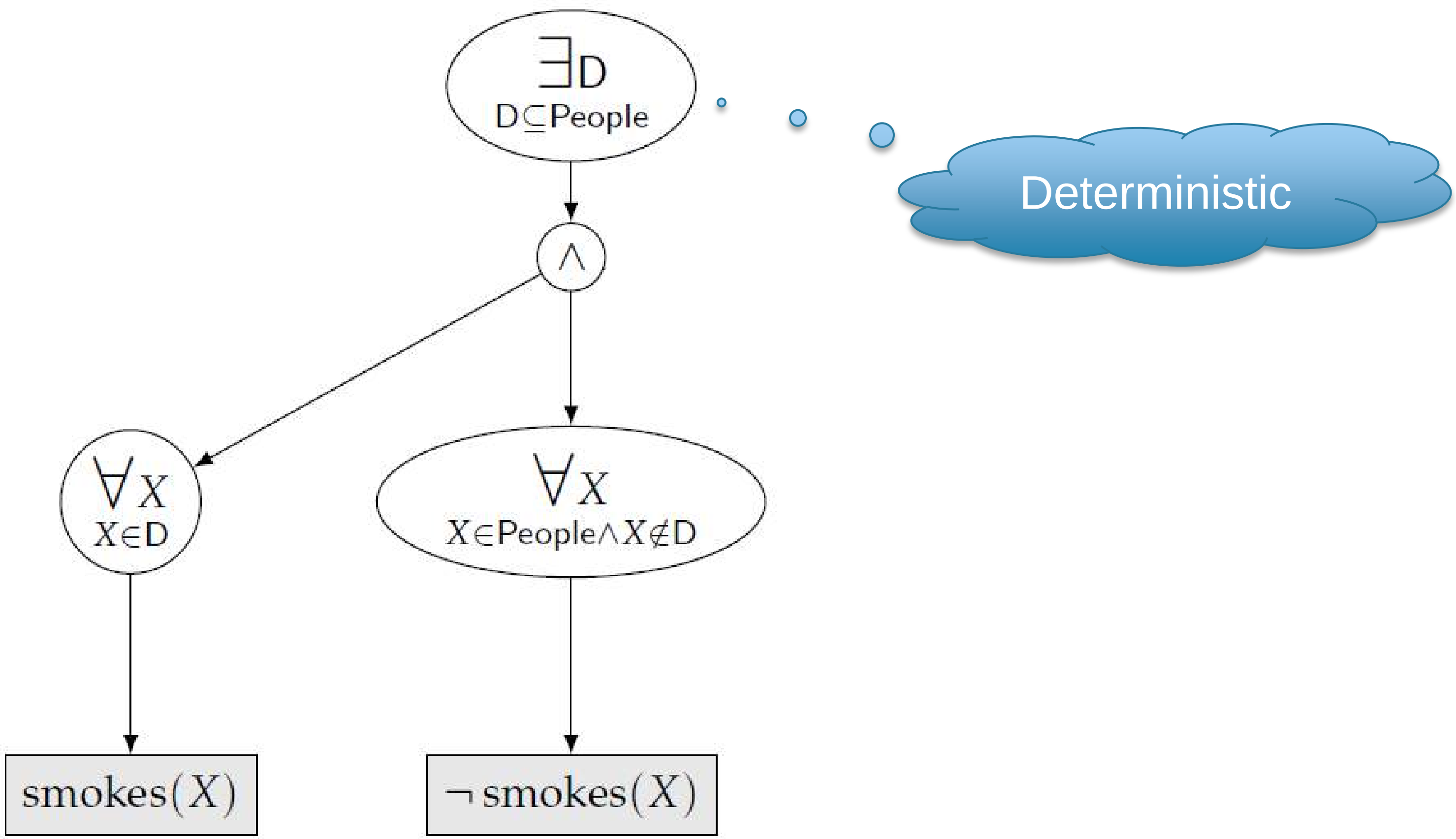
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[Van den Broeck 2013]

Deterministic Decomposable FO NNF

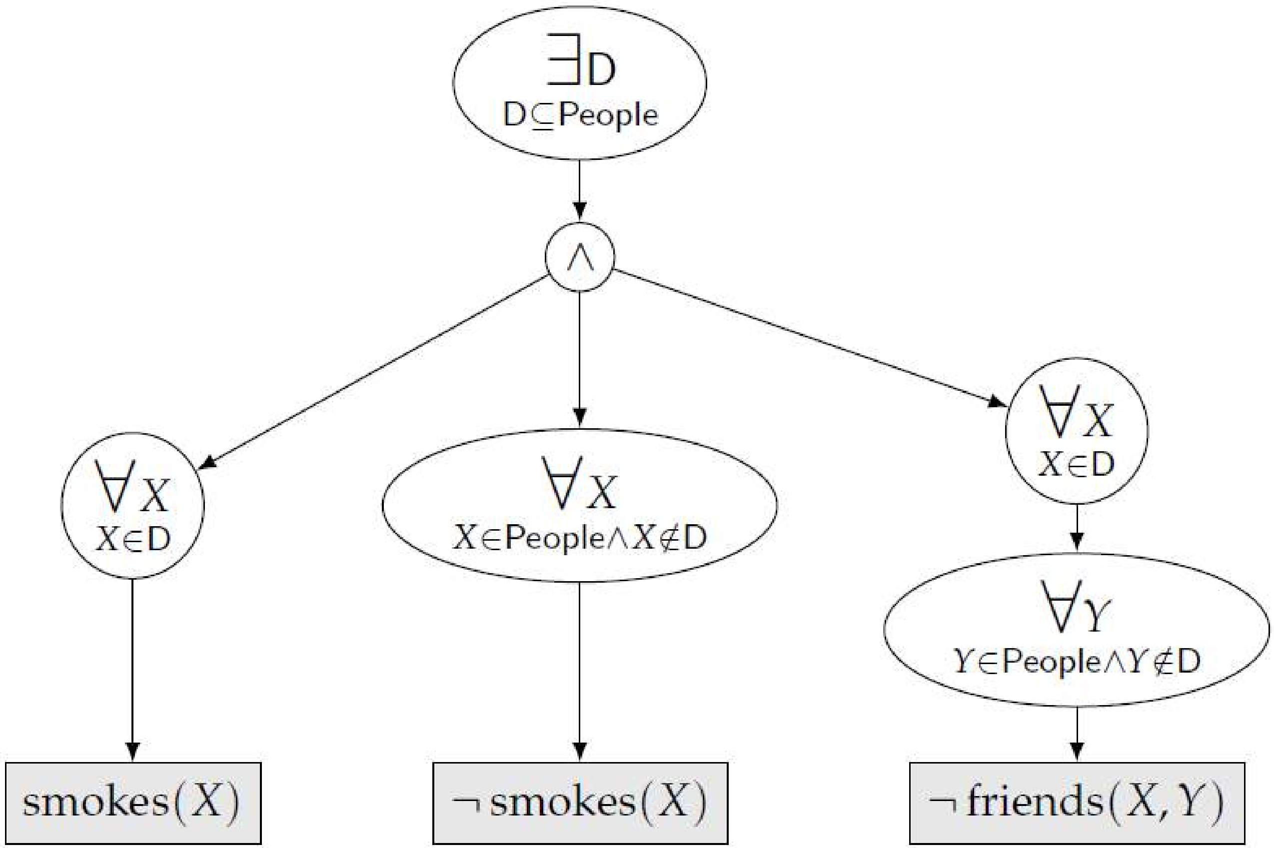
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[Van den Broeck 2013]

Deterministic Decomposable FO NNF

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)



[Van den Broeck 2013]

First-Order Model Counting: Example

$\Delta = \forall x, y \in$ **People**: $Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y)$

[Van den Broeck 2015]

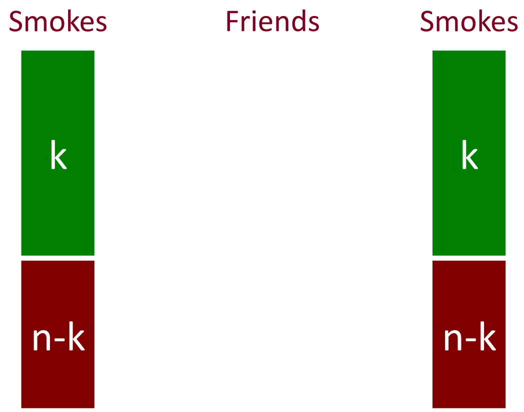
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

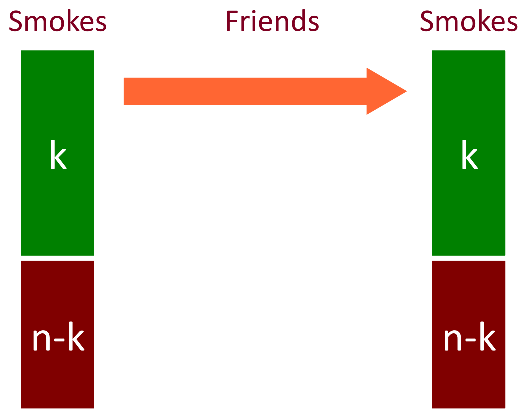
First-Order Model Counting: Example

$\Delta = \forall x, y \in$ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

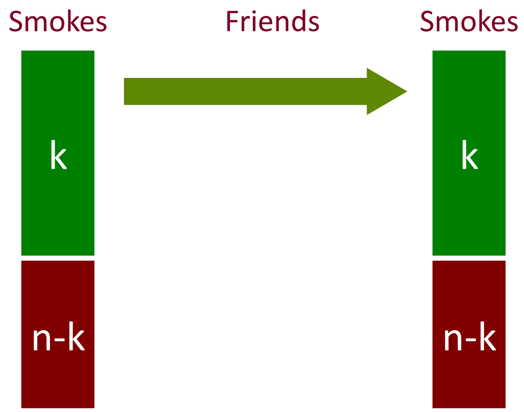
First-Order Model Counting: Example

Δ = ∀x ,y ∈ People: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

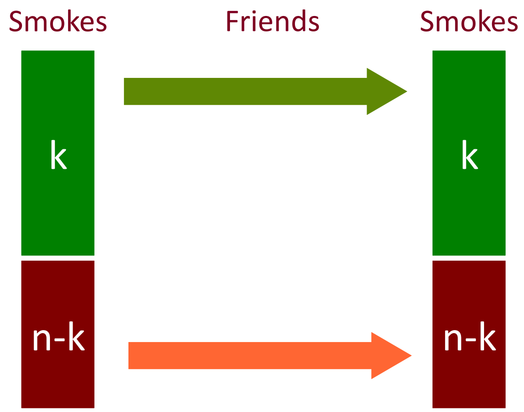
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

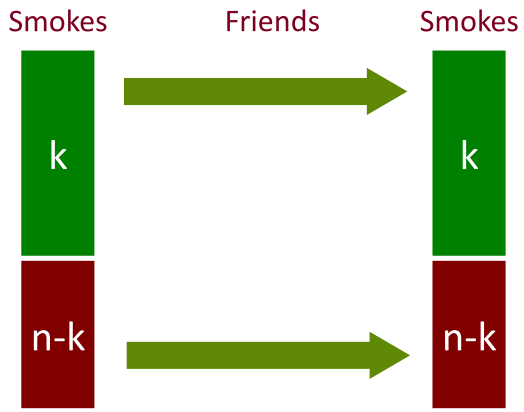
First-Order Model Counting: Example

$\Delta = \forall x, y \in$ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

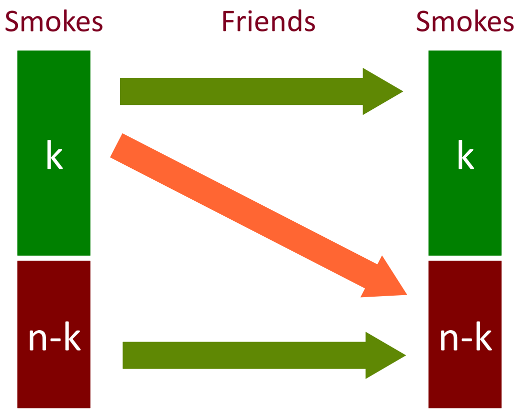
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

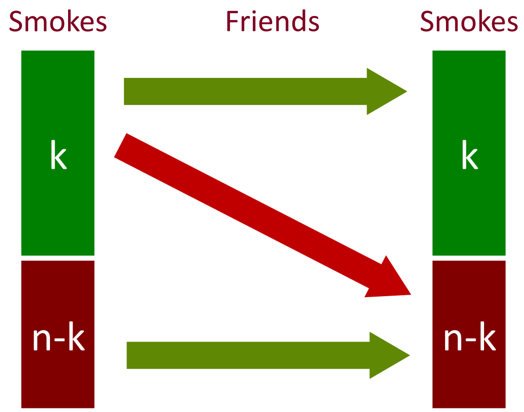
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

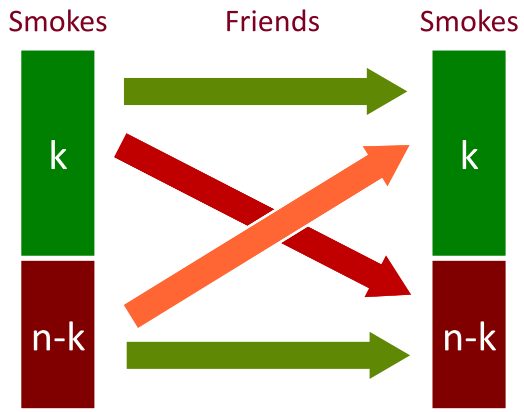
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

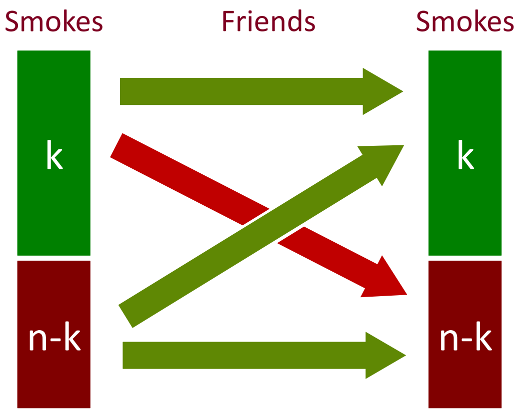
First-Order Model Counting: Example

Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



[Van den Broeck 2015]

First-Order Model Counting: Example

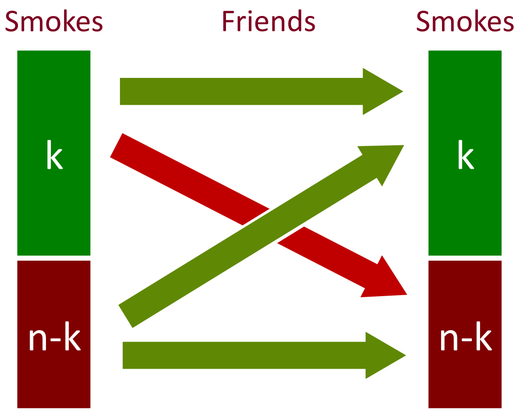
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

→ $2^{n^2 - k(n-k)}$ models



[Van den Broeck 2015]

First-Order Model Counting: Example

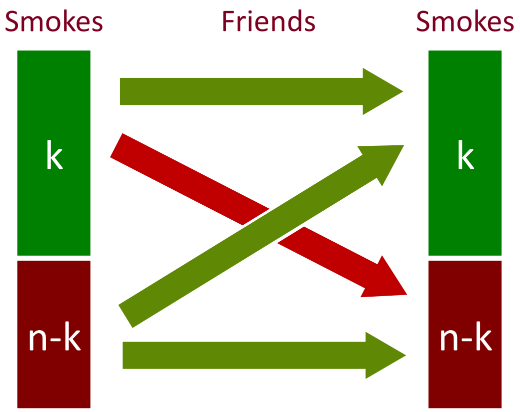
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

→ $2^{n^2-k(n-k)}$ models



- If we know that there are *k* smokers?

[Van den Broeck 2015]

First-Order Model Counting: Example

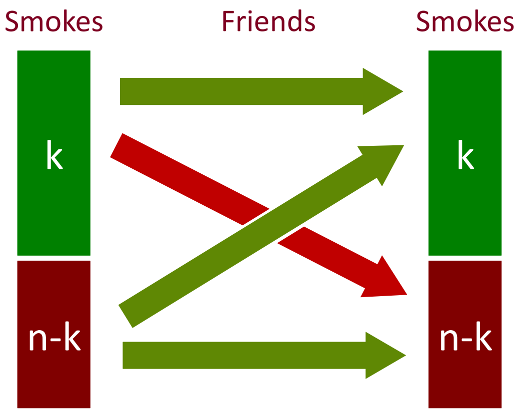
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

→ $2^{n^2-k(n-k)}$ models



- If we know that there are *k* smokers?

→ $\binom{n}{k} 2^{n^2-k(n-k)}$ models

[Van den Broeck 2015]

First-Order Model Counting: Example

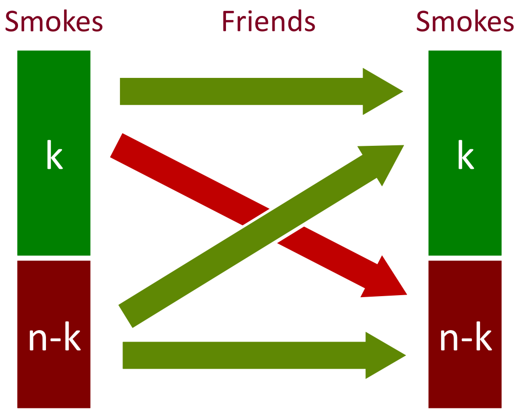
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

→ $2^{n^2-k(n-k)}$ models



- If we know that there are *k* smokers? → $\binom{n}{k} 2^{n^2-k(n-k)}$ models

- In total…

[Van den Broeck 2015]

First-Order Model Counting: Example

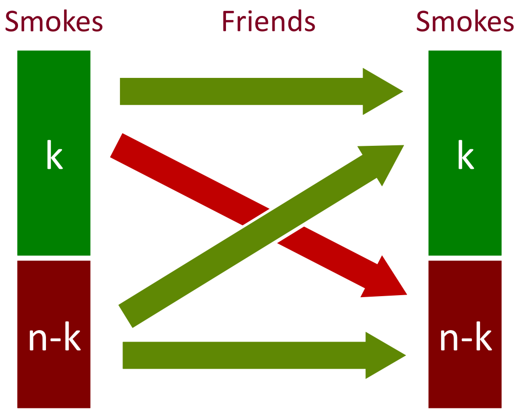
Δ = ∀x ,y ∈ **People**: Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

- If we know **D** precisely: who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

→ $2^{n^2 - k(n-k)}$ models



- If we know that there are *k* smokers? → $\binom{n}{k} 2^{n^2 - k(n-k)}$ models

- In total… → $\sum_{k=0}^{n} \binom{n}{k} 2^{n^2 - k(n-k)}$ models

[Van den Broeck 2015]

Main Positive Result: FO^2

- FO^2 = FO restricted to two variables
- "The graph has a path of length 10":

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

- Theorem: Compilation algorithm to FO d-DNNF is complete for FO^2
- Model counting for FO^2 in PTIME domain complexity

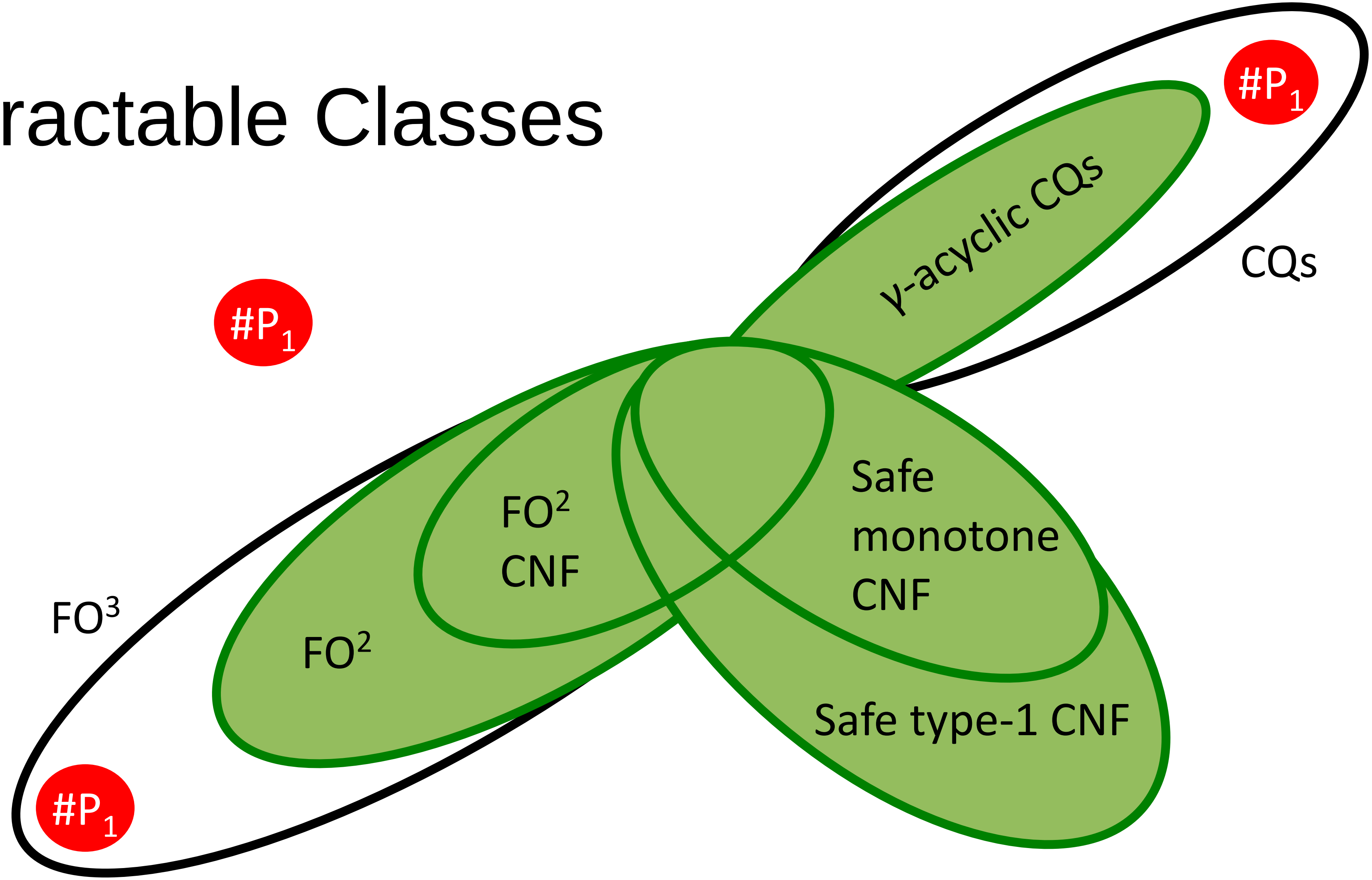
Main Negative Results

Domain complexity:

- There is an FO formula Q s.t. FOMC(Q, n) is **#P₁**-hard
- There is a Q in FO^3 s.t. FOMC(Q, n) is **#P₁**-hard
- There exists a conjunctive query Q s.t. symmetric WFOMC(Q, n) is **#P₁**-hard
- There exists a positive clause Q w.o. '=' s.t. symmetric WFOMC(Q, n) is **#P₁**-hard

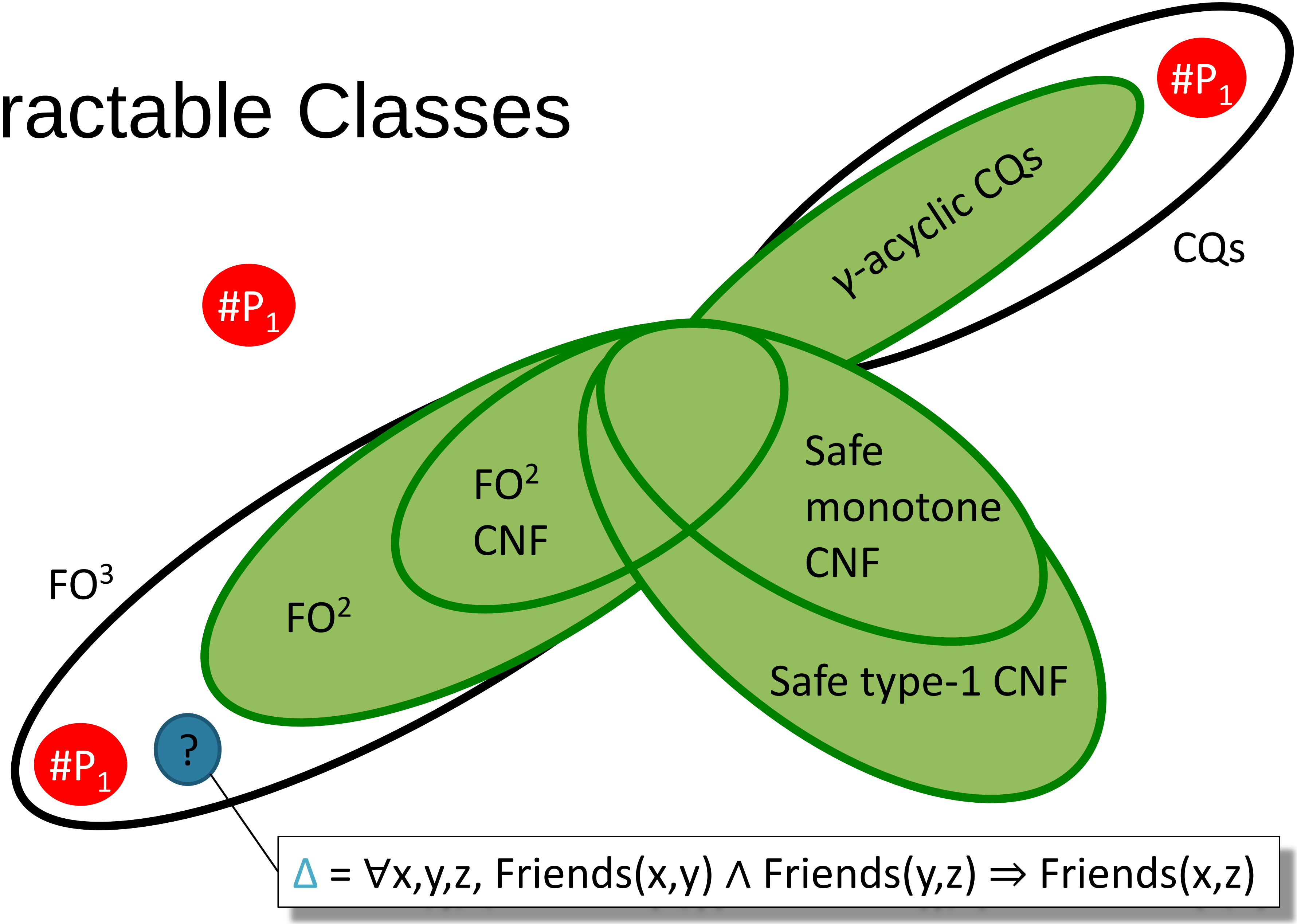
Therefore, no FO d-DNNF exists (unless…) ☹

Tractable Classes



[VdB; NIPS'11], [VdB et al.; KR'14], [Gribkoff, VdB, Suciu; UAI'15], [Beame, VdB, Gribkoff, Suciu; PODS'15], etc.

Tractable Classes



[VdB; NIPS'11], [VdB et al.; KR'14], [Gribkoff, VdB, Suciu; UAI'15], [Beame, VdB, Gribkoff, Suciu; PODS'15], etc.

Skolemization for WFOMC

$\Delta = \forall p, \exists c, Card(p,c)$

[VdB'14]

Skolemization for WFOMC

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

[VdB'14]

Skolemization for WFOMC

$\Delta = \forall p, \exists c, Card(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, Card(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

[VdB'14]

Skolemization for WFOMC

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\exists c, \text{Card}(p,c) = \text{true}$

$\exists c, \text{Card}(p,c) = \text{false}$

[VdB'14]

Skolemization for WFOMC

$\Delta = \forall p, \exists c, Card(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, Card(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\exists c, Card(p,c) = true$

$S(p) = true$

Also model of Δ, weight * 1

$\exists c, Card(p,c) = false$

[VdB'14]

Skolemization for WFOMC

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\exists c, \text{Card}(p,c)$ = true

- S(p) = true — Also model of Δ, weight * 1

$\exists c, \text{Card}(p,c)$ = false

- S(p) = true — No model of Δ, weight * 1
- S(p) = false — No model of Δ, weight * -1

Extra models

Cancel out

[VdB'14]

Resolution for WFOMC

$\Delta = \forall x \forall y\ (R(x) \lor \neg S(x,y)) \land \forall x \forall y\ (S(x,y) \lor T(y))$

Rules stuck…

Resolution on S(x,y): $\forall x \forall y\ (R(x) \lor T(y))$

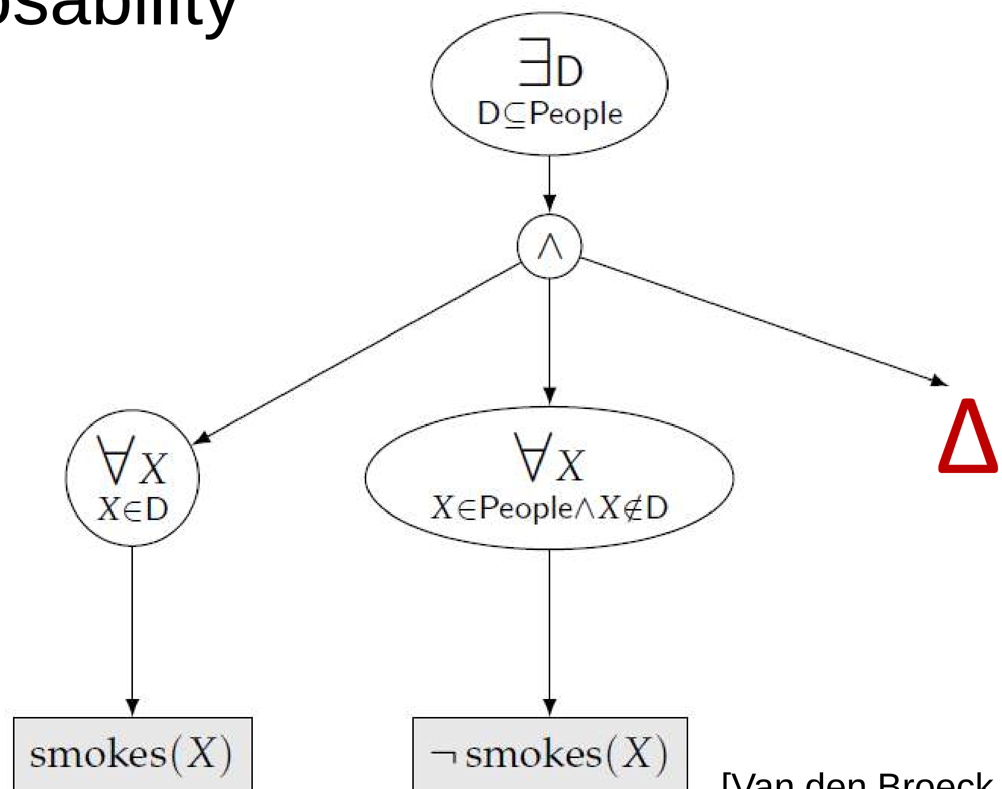
⇩

Add resolvent: $\Delta = \forall x \forall y\ (R(x) \lor \neg S(x,y)) \land \forall x \forall y\ (S(x,y) \lor T(y)) \land \forall x \forall y\ (R(x) \lor T(y))$

Now apply I/E!

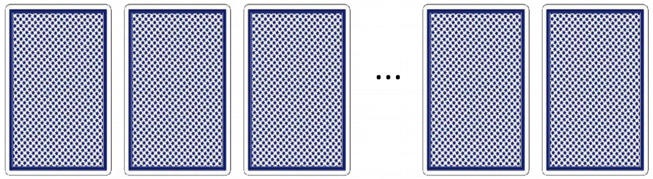
Compilation Rules

- Standard rules
  - Shannon decomposition (DPLL)
  - Detect decomposability
  - Etc.
- FO Shannon decomposition:



[Van den Broeck 2013]

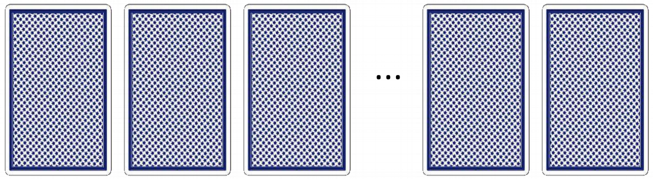
Playing Cards Revisited



$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \land Card(p,c') \Rightarrow c = c'$$

[Van den Broeck.; AAAI-KR'15]

Playing Cards Revisited



∀p, ∃c, Card(p,c)

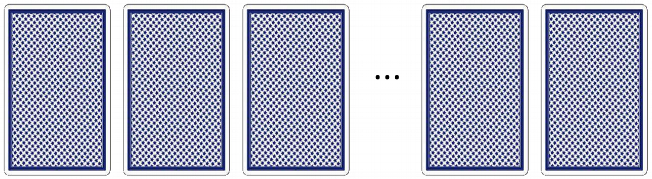
∀c, ∃p, Card(p,c)

∀p, ∀c, ∀c’, Card(p,c) ∧ Card(p,c’) ⇒ c = c’

$$\#SAT = \sum_{k=0}^{n} \binom{n}{k} \sum_{l=0}^{n} \binom{n}{l} (l+1)^k (-1)^{2n-k-l} = n!$$

[Van den Broeck.; AAAI-KR’15]

Playing Cards Revisited



∀p, ∃c, Card(p,c)
∀c, ∃p, Card(p,c)
∀p, ∀c, ∀c', Card(p,c) ∧ Card(p,c') ⇒ c = c'

$$\#SAT = \sum_{k=0}^{n} \binom{n}{k} \sum_{l=0}^{n} \binom{n}{l} (l+1)^k (-1)^{2n-k-l} = n!$$

Computed in time polynomial in n

[Van den Broeck.; AAAI-KR'15]

Overview

1. Propositional Refresher
2. Primer: A First-Order Tractable Language
3. Probabilistic Databases
4. Symmetric First-Order Model Counting
5. **Lots of Pointers**

Pointers

- Work on first-order knowledge compilation in `90s

Henry Kautz

- Factored Databases

Dan Olteanu

- New inference rules for symmetric counting (domain recursion)

Guy

More Pointers

- PTIME UCQ queries and circuit lower bounds

Paul Beame

- Compiling first-order database queries to propositional circuits

Dan Olteanu **Dan Suciu**

Pierre Bourhis **Pierre Senellart**

More Pointers

- Database fixed-parameter tractability

Antoine Amarilli **Guy**

- Colour refinement to detect first-order structure

Martin Grohe

- Probabilistic database preference models and triangle queries

Batya Kenig

Statistical Relational Learning

Markov Logic

3.14	Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic: 3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic: 3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

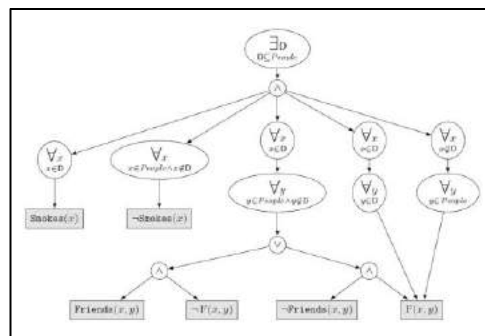
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Compile?

First-Order d-DNNF Circuit



[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic

3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

Domain

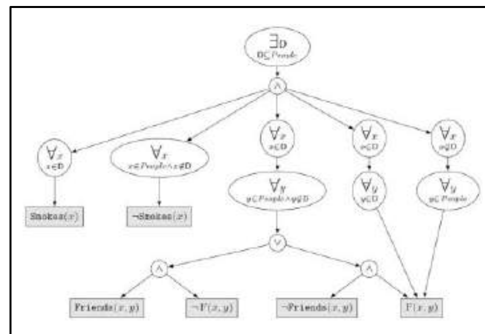
Alice
Bob
Charlie

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Compile?

First-Order d-DNNF Circuit



[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic

3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

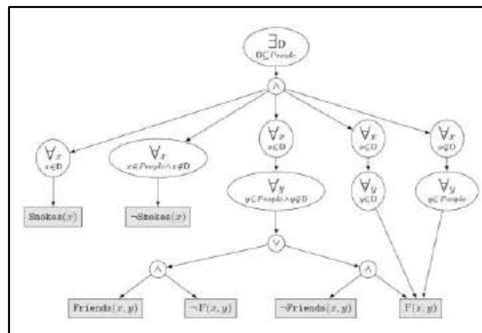
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Compile?

First-Order d-DNNF Circuit



Domain

Alice
Bob
Charlie

Z = WFOMC = 1479.85

[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic: 3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

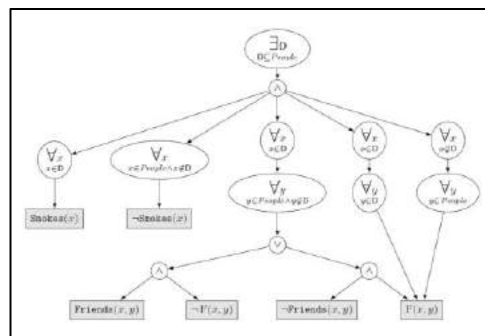
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Compile?

First-Order d-DNNF Circuit



Domain

Alice
Bob
Charlie

Z = WFOMC = 1479.85

Evaluation in time polynomial in domain size!

[Van den Broeck,PhD'13]

Statistical Relational Learning

Markov Logic: 3.14 Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)

Weight Function

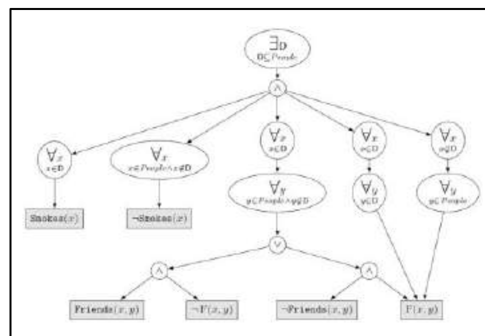
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=3.14
w(¬F)=1

FOL Sentence

∀x,y, F(x,y) ⇔ [Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y)]

Compile?

First-Order d-DNNF Circuit



Guy

Domain

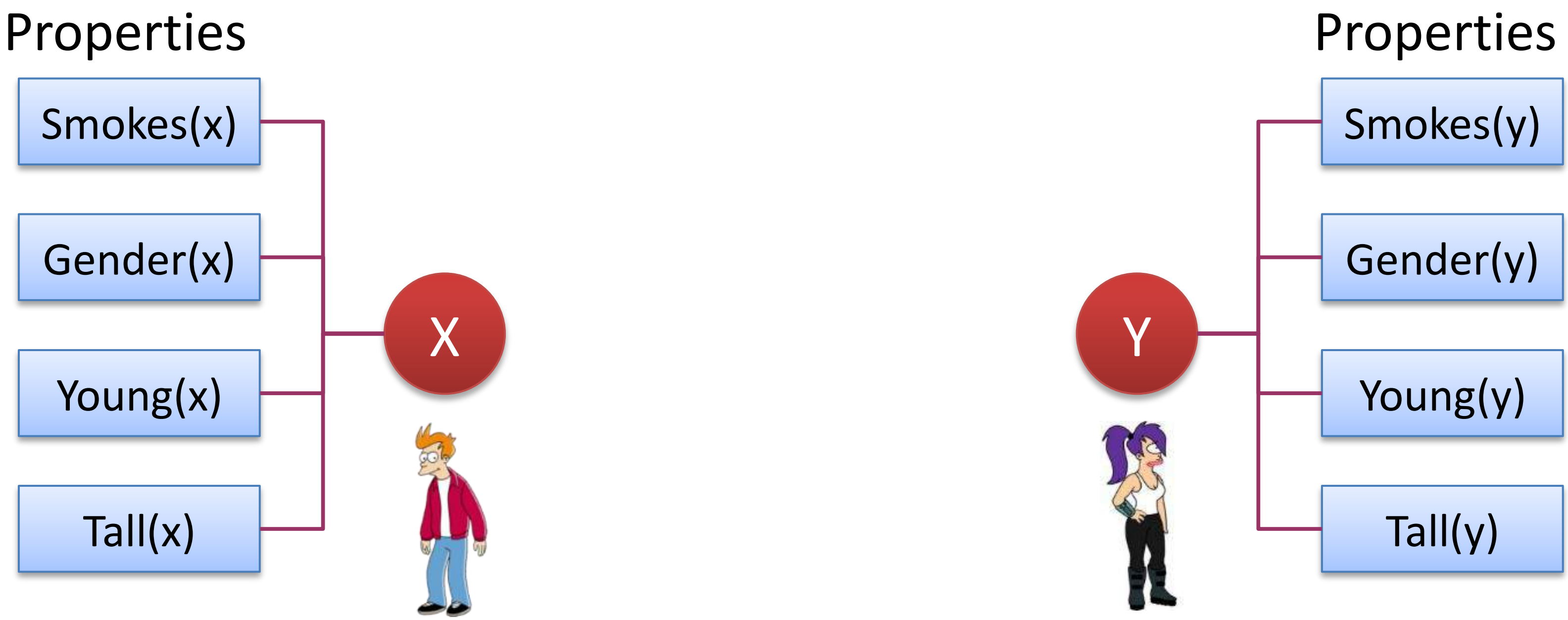
Alice
Bob
Charlie

Z = WFOMC = 1479.85

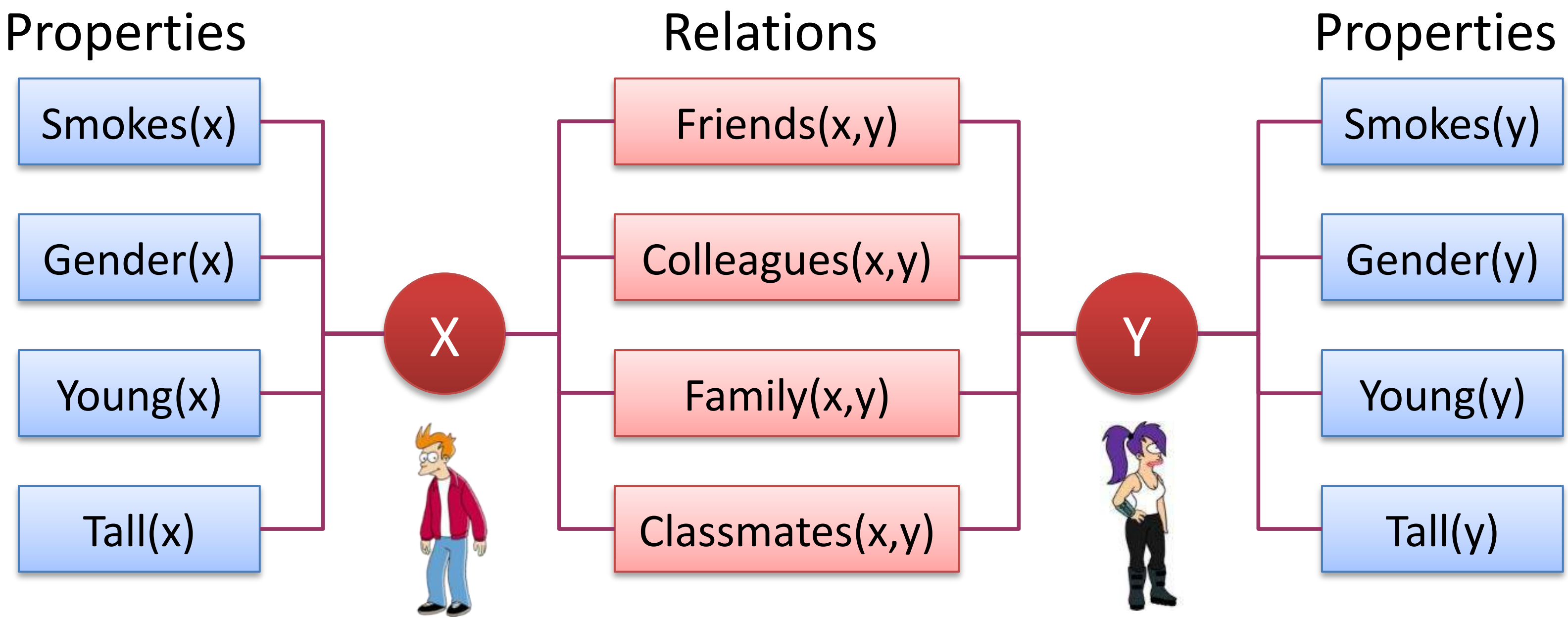
Evaluation in time polynomial in domain size!

[Van den Broeck,PhD'13]

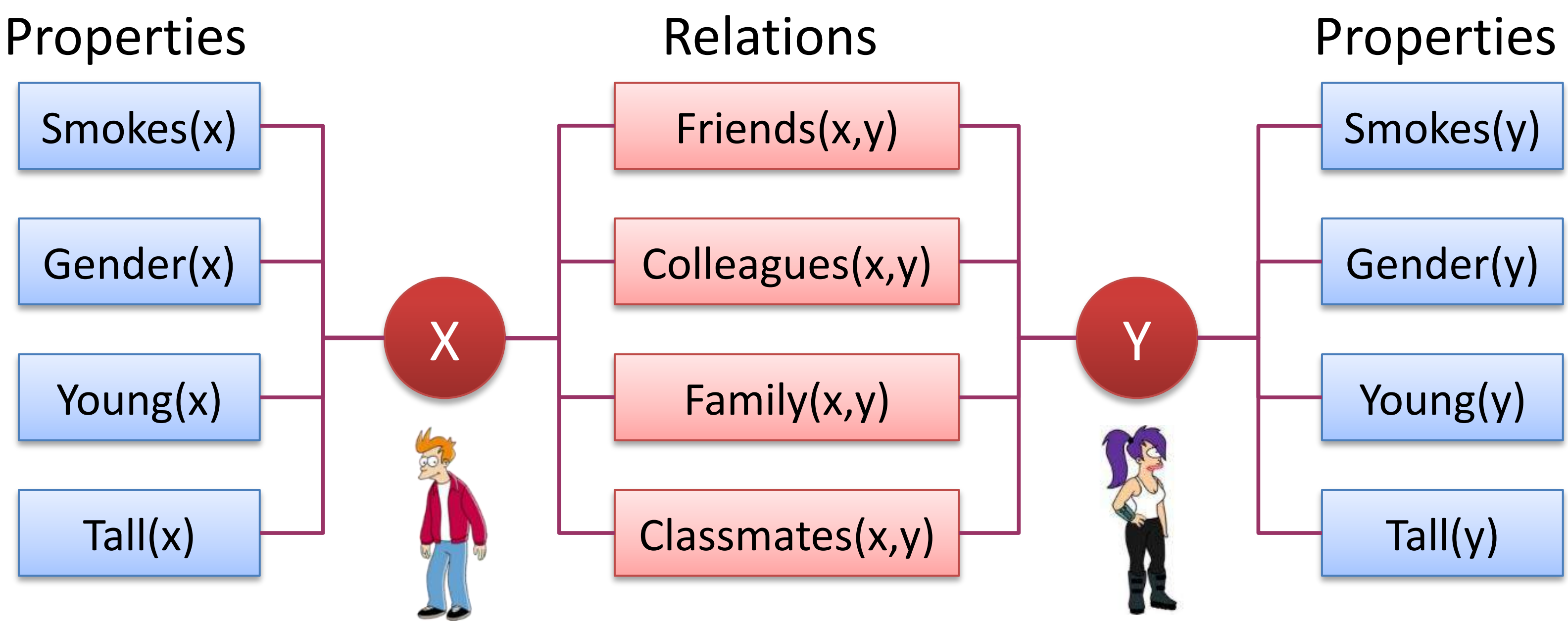
FO^2 is liftable!



FO^2 is liftable!



FO^2 is liftable!



"Smokers are more likely to be friends with other smokers."
"Colleagues of the same age are more likely to be friends."
"People are either family or friends, but never both."
"If X is family of Y, then Y is also family of X."
"If X is a parent of Y, then Y cannot be a parent of X."

More Pointers

- Lifted machine learning

- Open-world probabilistic databases

Generalized Model Counting

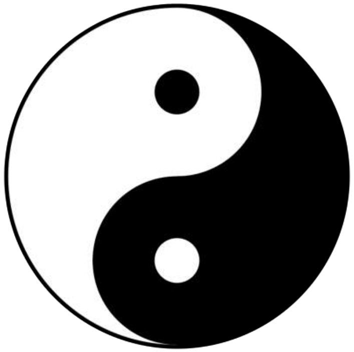
Probability Distribution

=

Logic

+

Weights



Generalized Model Counting

Probability Distribution	Logical Syntax
=	Model-theoretic Semantics
Logic	+
+	Weight function w(.)
Weights	

Weighted Model Integration

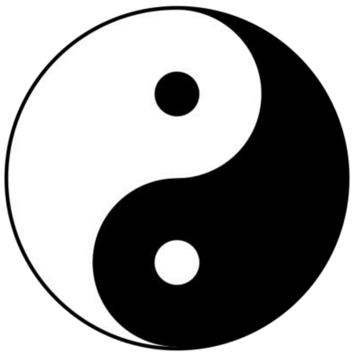
Probability Distribution

=

SMT(LRA)

+

Weights



[Belle et al. IJCAI'15, UAI'15]

Weighted Model Integration

Probability Distribution

=

SMT(LRA)

+

Weights

0 ≤ height ≤ 200
0 ≤ weight ≤ 200
0 ≤ age ≤ 100
age < 1 ⇒
height+weight ≤ 90

+

w(height))=height-10
w(¬height)=3*height^2
w(¬weight)=5
…

[Belle et al. IJCAI’15, UAI’15]

Weighted Model Integration

Probability Distribution

=

SMT

+

Weights

0 ≤ height ≤ 200
0 ≤ weight ≤ 200
0 ≤ age ≤ 100
age ≤ 1 →
weight ≤ 90

+

w(height))=height-10
w(¬height)=3*height^2
w(¬weight)=5
…

Scott Sanner

[Belle et al. IJCAI'15, UAI'15]

Probabilistic Programming

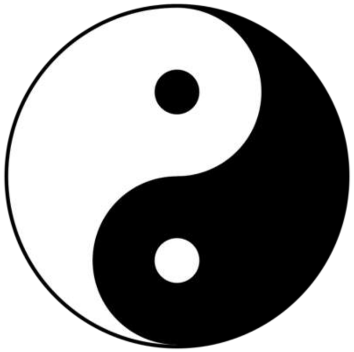
Probability Distribution

=

Logic Programs

+

Weights



[Fierens et al., TPLP'15]

Probabilistic Programming

Probability Distribution

=

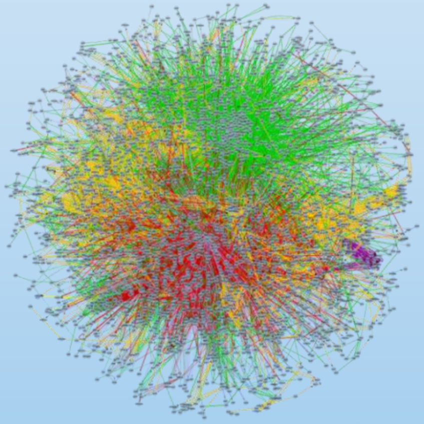
Logic Programs

+

Weights

```
path(X,Y) :-
        edge(X,Y).
path(X,Y) :-
        edge(X,Z), path(Z,Y).
```

+



[Fierens et al., TPLP'15]

Probabilistic Programming

Probability Distribution

=

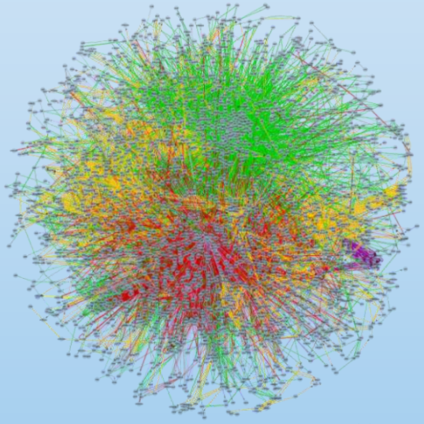
Logic P

+

Weights

```
path(X,Y) :-
        edge(X,Y).
path(X,Y) :-
        edge(X,Z), path(Z,Y).
```

Wannes Meert



[Fierens et al., TPLP’15]

Conclusions

- Determinism and decomposability generalize to first-order logic
- First-order model counting unifies
  - Probabilistic databases
  - High-level statistical AI models
- Fascinating computational complexity questions
- Requires dedicated first-order solvers

QUESTIONS?