Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: Completeness of Lifted Inference
- Part 6: Query Compilation
- Part 7: Symmetric Lifted Inference Complexity
- Part 8: Open-World Probabilistic Databases
- Part 9: Discussion & Conclusions

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc.

- A formal definition: **Domain-lifted inference**

> Inference runs in time polynomial
> in the number of objects in the domain.

  - Polynomial in #people, #webpages, #cards
  - Not polynomial in #predicates, #formulas, #logical variables
  - Related to data complexity in databases

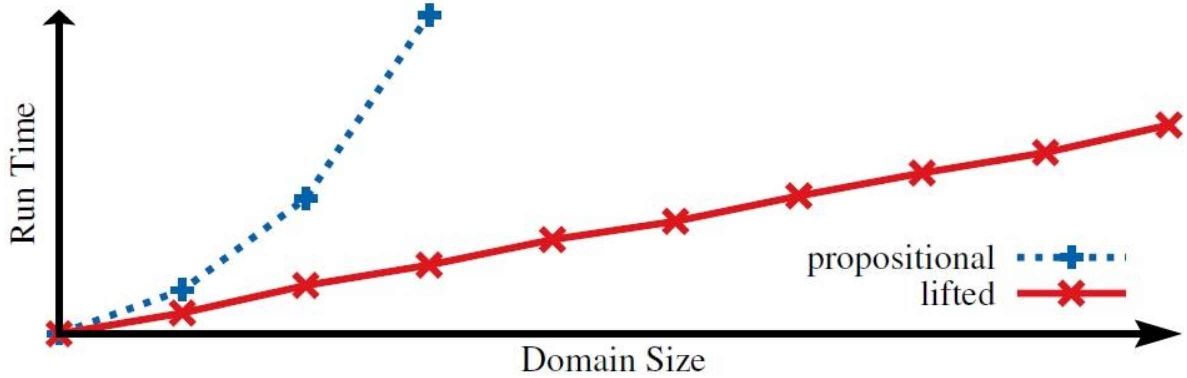
[VdB'11, Jaeger'12]

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc. [Poole'03, etc.]

- A formal definition: **Domain-lifted inference**



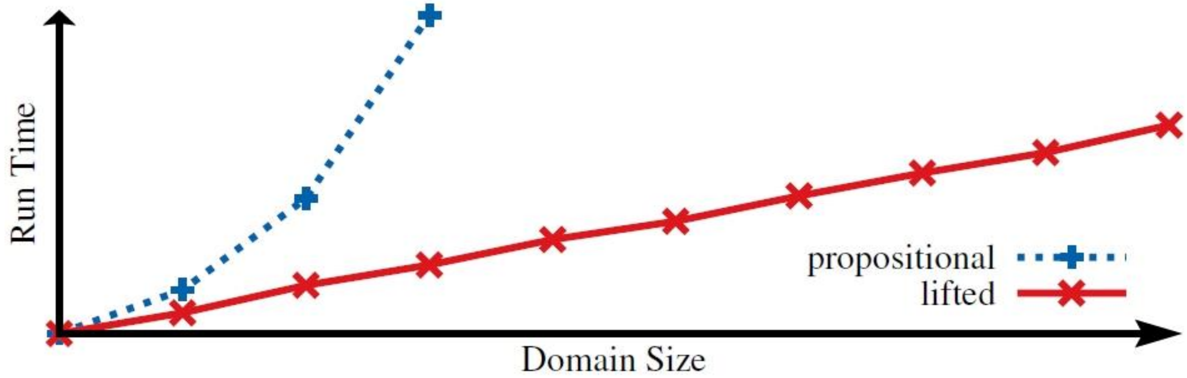
[VdB'11, Jaeger'12]

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc. [Poole'03, etc.]

- A formal definition: **Domain-lifted inference**



- Alternative in this tutorial:

  Lifted inference = ∃Query Plan = ∃FO Compilation

[VdB'11, Jaeger'12]

Asymmetric WFOMC Rules

Preprocess Q (omitted from this talk; see [Suciu'11]), then apply these rules (some have preconditions)

$\boxed{\mathrm{WMC}(\neg\Delta) = Z - \mathrm{WMC}(\Delta)}$ Negation

Normalization constant Z
(easy to compute)

Asymmetric WFOMC Rules

Preprocess Q (omitted from this talk; see [Suciu'11]), then apply these rules (some have preconditions)

$$\mathrm{WMC}(\neg\Delta) = Z - \mathrm{WMC}(\Delta)$$

Negation

Normalization constant Z (easy to compute)

$$\mathrm{WMC}(\Delta_1 \wedge \Delta_2) = \mathrm{WMC}(\Delta_1) * \mathrm{WMC}(\Delta_2)$$
$$\mathrm{WMC}(\Delta_1 \vee \Delta_2) = Z - (Z_1 - \mathrm{WMC}(\Delta_1)) * (Z_2 - \mathrm{WMC}(\Delta_2))$$

Independent join / union

Asymmetric WFOMC Rules

Preprocess Q (omitted from this talk; see [Suciu'11]), then apply these rules (some have preconditions)

$\mathrm{WMC}(\neg\Delta) = Z - \mathrm{WMC}(\Delta)$

Negation

Normalization constant Z (easy to compute)

$\mathrm{WMC}(\Delta_1 \wedge \Delta_2) = \mathrm{WMC}(\Delta_1) * \mathrm{WMC}(\Delta_2)$

$\mathrm{WMC}(\Delta_1 \vee \Delta_2) = Z - (Z_1 - \mathrm{WMC}(\Delta_1)) * (Z_2 - \mathrm{WMC}(\Delta_2))$

Independent join / union

$\mathrm{WMC}(\exists z\, \Delta) = Z - \Pi_{C \in \mathrm{Domain}} (Z_C - \mathrm{WMC}(\Delta[C/z])$

$\mathrm{WMC}(\forall z\, \Delta) = \Pi_{C \in \mathrm{Domain}} \mathrm{WMC}(\Delta[C/z]$

Independent project

Asymmetric WFOMC Rules

Preprocess Q (omitted from this talk; see [Suciu'11]), then apply these rules (some have preconditions)

$\mathrm{WMC}(\neg\Delta) = Z\text{-}\mathrm{WMC}(\Delta)$

Negation

Normalization constant Z
(easy to compute)

$\mathrm{WMC}(\Delta_1 \wedge \Delta_2) = \mathrm{WMC}(\Delta_1) * \mathrm{WMC}(\Delta_2)$

$\mathrm{WMC}(\Delta_1 \vee \Delta_2) = Z - (Z_1 - \mathrm{WMC}(\Delta_1)) * (Z_2 - \mathrm{WMC}(\Delta_2))$

Independent join / union

$\mathrm{WMC}(\exists z\ \Delta) = Z - \Pi_{C \in \mathrm{Domain}} (Z_C - \mathrm{WMC}(\Delta[C/z]))$

$\mathrm{WMC}(\forall z\ \Delta) = \Pi_{C \in \mathrm{Domain}} \mathrm{WMC}(\Delta[C/z])$

Independent project

$\mathrm{WMC}(\Delta_1 \wedge \Delta_2) = \mathrm{WMC}(\Delta_1) + \mathrm{WMC}(\Delta_2) - \mathrm{WMC}(\Delta_1 \vee \Delta_2)$

$\mathrm{WMC}(\Delta_1 \vee \Delta_2) = \mathrm{WMC}(\Delta_1) + \mathrm{WMC}(\Delta_2) - \mathrm{WMC}(\Delta_1 \wedge \Delta_2)$

Inclusion/ exclusion

Symmetric WFOMC Rules

- Simplification to *independent project*:

If $\Delta[C_1/x]$, $\Delta[C_2/x]$, … are independent

$$\mathrm{WMC}(\exists z\ \Delta) = Z - (Z_{C_1} - \mathrm{WMC}(\Delta[C_1/z])^{|\mathrm{Domain}|}$$

$$\mathrm{WMC}(\forall z\ \Delta) = \mathrm{WMC}(\Delta[C_1/z])^{|\mathrm{Domain}|}$$

[VdB'11]

Symmetric WFOMC Rules

- Simplification to *independent project*:

  If $\Delta[C_1/x]$, $\Delta[C_2/x]$, … are independent

  $\mathrm{WMC}(\exists z\ \Delta) = Z - (Z_{C_1} - \mathrm{WMC}(\Delta[C_1/z])^{|\mathrm{Domain}|}$

  $\mathrm{WMC}(\forall z\ \Delta) = \mathrm{WMC}(\Delta[C_1/z])^{|\mathrm{Domain}|}$

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

The workhorse of
Symmetric WFOMC

[VdB'11]

WFOMC Inference: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

WFOMC Inference: Example

- FO-Model Counting: w(R) = w(¬R) = 1
- Apply inference rules backwards (step 4-3-2-1)

4. Δ = (Stress(Alice) ⇒ Smokes(Alice))

Domain = {Alice}

WFOMC Inference: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

---

4. $\boxed{\Delta = (\text{Stress(Alice)} \Rightarrow \text{Smokes(Alice)})}$

$\boxed{\text{Domain} = \{\text{Alice}\}}$

→ 3 models

WFOMC Inference: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

---

4. $\boxed{\Delta = (\text{Stress(Alice)} \Rightarrow \text{Smokes(Alice)})}$ $\boxed{\text{Domain} = \{\text{Alice}\}}$

$$\begin{aligned}
&\text{WMC}(\neg\text{Stress(Alice)} \lor \text{Smokes(Alice)}) = \\
&\quad = Z - \text{WMC}(\text{Stress(Alice)}) \times \text{WMC}(\neg\text{Smokes(Alice)}) \\
&\quad = 4 - 1 \times 1 = 3 \text{ models}
\end{aligned}$$

WFOMC Inference: Example

- FO-Model Counting: w(R) = w(¬R) = 1
- Apply inference rules backwards (step 4-3-2-1)

---

4. Δ = (Stress(Alice) ⇒ Smokes(Alice))

Domain = {Alice}

WMC(¬Stress(Alice) ∨ Smokes(Alice))) =
= Z – WMC(Stress(Alice)) × WMC(¬Smokes(Alice))
= 4 – 1 × 1 = 3 models

---

3. Δ = ∀x, (Stress(x) ⇒ Smokes(x))

Domain = {n people}

WFOMC Inference: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

4. $\Delta = (\text{Stress(Alice)} \Rightarrow \text{Smokes(Alice)})$

Domain = {Alice}

$$\text{WMC}(\neg\text{Stress(Alice)} \lor \text{Smokes(Alice)})) =$$
$$= Z - \text{WMC}(\text{Stress(Alice)}) \times \text{WMC}(\neg\text{Smokes(Alice)})$$
$$= 4 - 1 \times 1 = 3 \text{ models}$$

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$

Domain = {n people}

$\rightarrow 3^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$

Domain = {n people}

$\rightarrow 3^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$

Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))$

D = {n people}

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$

Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (ParentOf(y) \wedge Female \Rightarrow MotherOf(y))$

D = {n people}

If Female = true? $\Delta = \forall y, (ParentOf(y) \Rightarrow MotherOf(y))$ $\rightarrow 3^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$

Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (ParentOf(y) \land Female \Rightarrow MotherOf(y))$

D = {n people}

If Female = true? $\Delta = \forall y, (ParentOf(y) \Rightarrow MotherOf(y))$ $\rightarrow 3^n$ models

If Female = false? Δ = true $\rightarrow 4^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$

Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (ParentOf(y) \wedge Female \Rightarrow MotherOf(y))$

D = {n people}

If Female = true? $\Delta = \forall y, (ParentOf(y) \Rightarrow MotherOf(y))$ $\rightarrow 3^n$ models

If Female = false? Δ = true $\rightarrow 4^n$ models

$\rightarrow 3^n + 4^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$ Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (ParentOf(y) \wedge Female \Rightarrow MotherOf(y))$ D = {n people}

$WMC(\Delta) = WMC(\neg Female \vee \forall y, (ParentOf(y) \Rightarrow MotherOf(y)))$
$= 2 * 2^n * 2^n - (2 - 1) * (2^n * 2^n - WMC(\forall y, (ParentOf(y) \Rightarrow MotherOf(y))))$
$= 2 * 4^n - (4^n - 3^n)$

$\rightarrow 3^n + 4^n$ models

WFOMC Inference: Example

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$

Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (\text{ParentOf}(y) \land \text{Female} \Rightarrow \text{MotherOf}(y))$

D = {n people}

$$\text{WMC}(\Delta) = \text{WMC}(\neg\,\text{Female} \lor \forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)))$$
$$= 2 * 2^n * 2^n - (2 - 1) * (2^n * 2^n - \text{WMC}(\forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y))))$$
$$= 2 * 4^n - (4^n - 3^n)$$

$\rightarrow 3^n + 4^n$ models

1. $\Delta = \forall x,y, (\text{ParentOf}(x,y) \land \text{Female}(x) \Rightarrow \text{MotherOf}(x,y))$

D = {n people}

WFOMC Inference: Example

3. $\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))$ Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (ParentOf(y) \land Female \Rightarrow MotherOf(y))$ D = {n people}

$$\begin{aligned} WMC(\Delta) &= WMC(\neg Female \lor \forall y, (ParentOf(y) \Rightarrow MotherOf(y))) \\ &= 2 * 2^n * 2^n - (2-1) * (2^n * 2^n - WMC(\forall y, (ParentOf(y) \Rightarrow MotherOf(y)))) \\ &= 2 * 4^n - (4^n - 3^n) \end{aligned}$$

$\rightarrow 3^n + 4^n$ models

1. $\Delta = \forall x,y, (ParentOf(x,y) \land Female(x) \Rightarrow MotherOf(x,y))$ D = {n people}

$\rightarrow (3^n + 4^n)^n$ models

Atom Counting: Example

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

Domain = {n people}

Atom Counting: Example

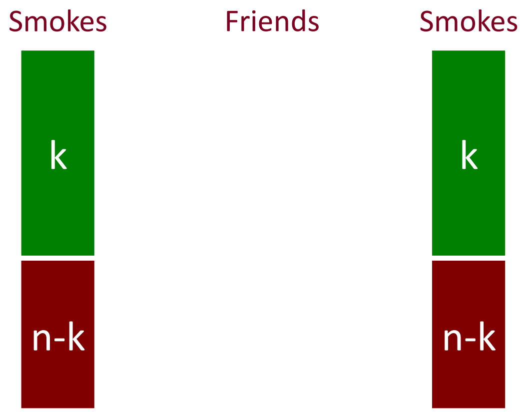
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

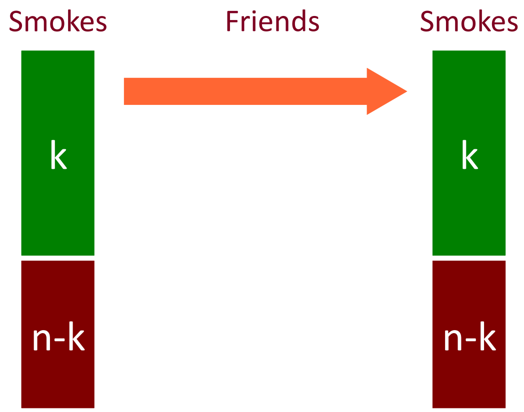
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

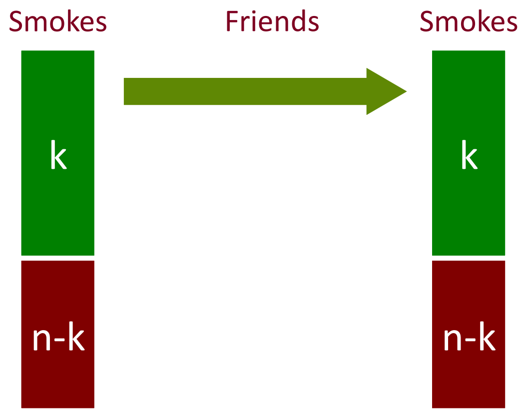
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

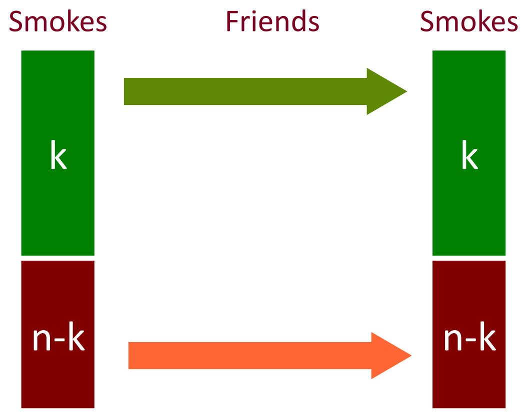
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

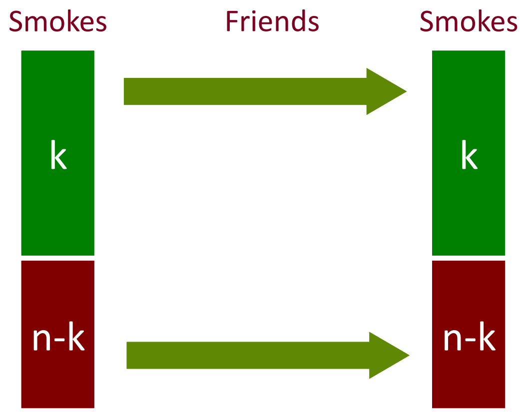
$\Delta = \forall x,y, (\text{Smokes}(x) \land \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

Domain = {n people}

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

Database:

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



Atom Counting: Example

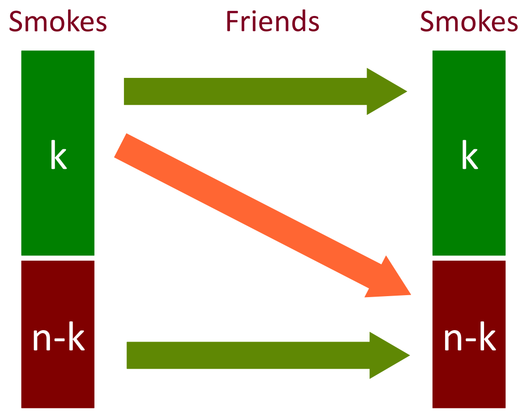
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

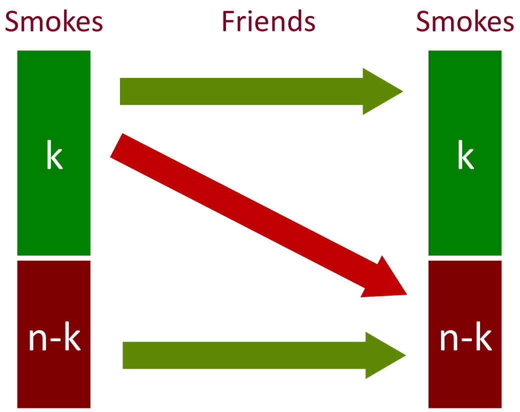
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

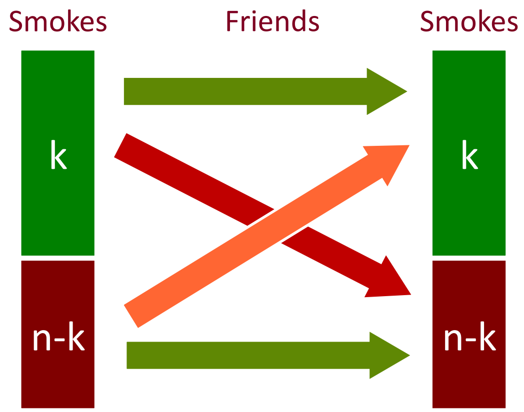
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

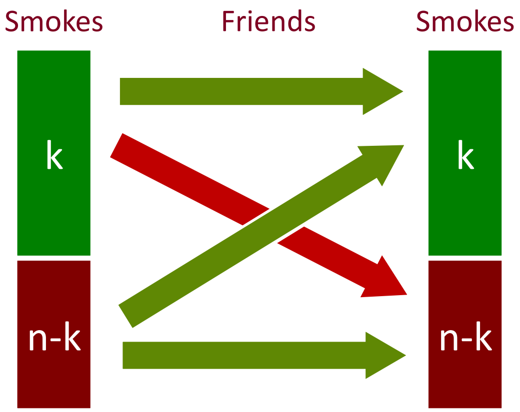
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

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

Database:

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



Atom Counting: Example

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

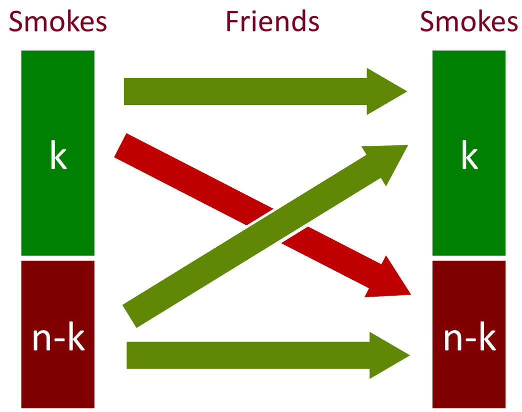
Domain = {n people}

- If we know 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



Atom Counting: Example

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

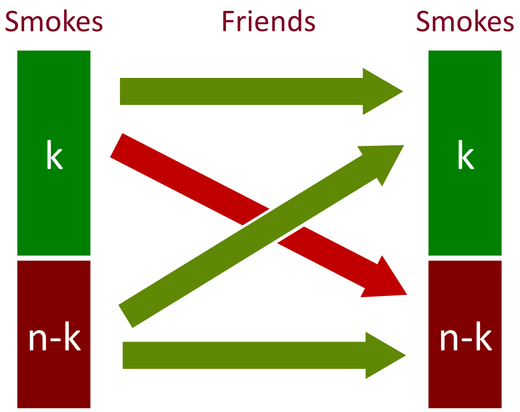
Domain = {n people}

- If we know 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?

Atom Counting: Example

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

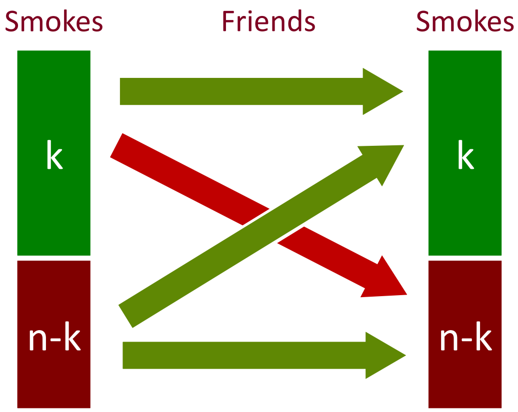
Domain = {n people}

- If we know 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

Atom Counting: Example

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

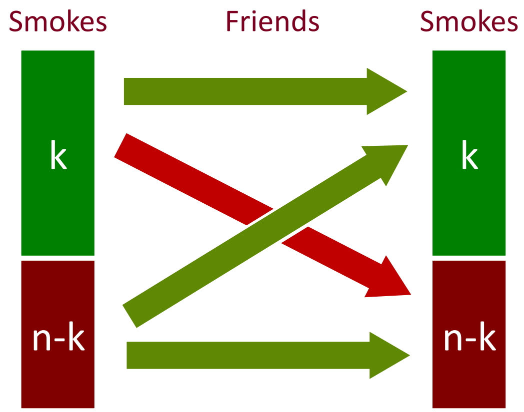
Domain = {n people}

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

Database:

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

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



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

- In total…

Atom Counting: Example

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

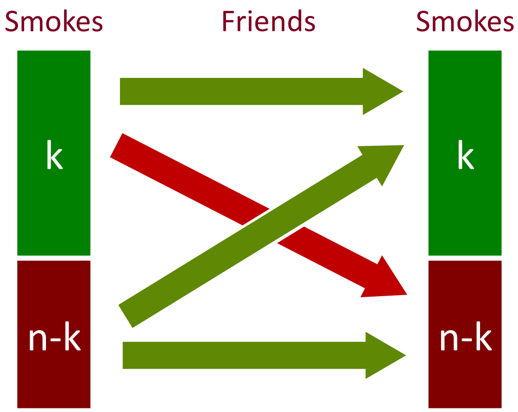
Domain = {n people}

- If we know 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

Augment Rules with Logical Rewritings

[Suciu'11]

Augment Rules with Logical Rewritings

1. Remove constants (shattering)

$\Delta = \forall x\ (\mathrm{Friend}(\mathrm{Alice}, x) \vee \mathrm{Friend}(x, \mathrm{Bob}))$

[Suciu'11]

Augment Rules with Logical Rewritings

1. Remove constants (shattering)

$\Delta = \forall x\ (\text{Friend}(\text{Alice}, x) \lor \text{Friend}(x, \text{Bob}))$

$\Rightarrow$ $\Delta = \forall x\ (F_1(x) \lor F_2(x)) \land (F_3 \lor F_4) \land (F_4 \lor F_5)$

$F_1(x) = \text{Friend}(\text{Alice},x)$
$F_2(x) = \text{Friend}(x,\text{Bob})$
$F_3 = \text{Friend}(\text{Alice}, \text{Alice})$
$F_4 = \text{Friend}(\text{Alice},\text{Bob})$
$F_5 = \text{Friend}(\text{Bob},\text{Bob})$

[Suciu'11]

Augment Rules with Logical Rewritings

$F_1(x)$ = Friend(Alice,x)
$F_2(x)$ = Friend(x,Bob)
F_3 = Friend(Alice, Alice)
F_4 = Friend(Alice,Bob)
F_5 = Friend(Bob,Bob)

1. Remove constants (shattering)

$\Delta = \forall x\ (\mathrm{Friend}(\mathrm{Alice}, x) \vee \mathrm{Friend}(x, \mathrm{Bob}))$

$\Rightarrow$ $\Delta = \forall x\ (F_1(x) \vee F_2(x)) \wedge (F_3 \vee F_4) \wedge (F_4 \vee F_5)$

2. "Rank" variables (= occur in the same order in each atom)

$\Delta = (\mathrm{Friend}(x,y) \vee \mathrm{Enemy}(x,y)) \quad \wedge \quad (\mathrm{Friend}(x,y) \vee \mathrm{Enemy}(y,x))$

Wrong order

[Suciu'11]

Augment Rules with Logical Rewritings

$F_1(x) = \text{Friend}(\text{Alice},x)$
$F_2(x) = \text{Friend}(x,\text{Bob})$
$F_3 = \text{Friend}(\text{Alice}, \text{Alice})$
$F_4 = \text{Friend}(\text{Alice},\text{Bob})$
$F_5 = \text{Friend}(\text{Bob},\text{Bob})$

1. Remove constants (shattering)

$\Delta = \forall x\ (\text{Friend}(\text{Alice}, x) \vee \text{Friend}(x, \text{Bob}))$

$\Rightarrow$ $\Delta = \forall x\ (F_1(x) \vee F_2(x)) \wedge (F_3 \vee F_4) \wedge (F_4 \vee F_5)$

2. "Rank" variables (= occur in the same order in each atom)

$\Delta = (\text{Friend}(x,y) \vee \text{Enemy}(x,y)) \quad \wedge \quad (\text{Friend}(x,y) \vee \text{Enemy}(y,x))$

Wrong order

$F_1(u,v) = \text{Friend}(u,v), u<v$
$F_2(u) \quad = \text{Friend}(u,u)$
$F_3(u,v) = \text{Friend}(v,u), v<u$

$E_1(u,v) = \text{Friend}(u,v), u<v$
$E_2(u) \quad = \text{Friend}(u,u)$
$E_3(u,v) = \text{Friend}(v,u), v<u$

$\Rightarrow$

$$\begin{aligned}\Delta = &(F_1(x,y) \vee E_1(x,y)) \wedge (F_1(x,y) \vee E_3(x,y))\\ &\wedge (F_2(x) \vee E_2(x))\\ &\wedge (F_3(x,y) \vee E_3(x,y)) \wedge (F_3(x,y) \vee E_1(x,y))\end{aligned}$$

[Suciu'11]

Augment Rules with Logical Rewritings

3. Perform Resolution [Gribkoff'14]

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

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!

Augment Rules with Logical Rewritings

4. Skolemization [VdB'14]

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

Inference rules assume one type of quantifier!

Mix ∀/∃ in encodings of MLNs with quantifiers and probabilistic programs

Datalog `smokes(X) :- friends(X,Y), smokes(Y).`

FOL $\Delta = \forall x, \text{Smokes}(x) \Leftrightarrow \exists y, \text{Friends}(x,y), \text{Smokes}(y).$

⇩

Skolemization Input: Mix ∀/∃ Output: Only ∀

BUT: cannot introduce Skolem constants or functions! $\forall p, \text{Card}(p,S(p))$

Skolemization: Example

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

[VdB’14]

Skolemization: Example

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

Skolemization

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

[VdB'14]

Skolemization: Example

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

Skolemization

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

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

Skolem predicate

[VdB'14]

Skolemization: Example

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

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

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

[VdB'14]

Skolemization: Example

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

Skolemization

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

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

Skolem predicate

Consider one position p:

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

$S(p) = \text{true}$

Also model of Δ, weight $* 1$

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

[VdB'14]

Skolemization: Example

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

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]

First-Order Knowledge Compilation

Markov Logic

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

[Vdb'11,'13]

First-Order Knowledge Compilation

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)=exp(3.14)
w(¬F)=1

FOL Sentence

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

[Vdb'11,'13]

First-Order Knowledge Compilation

Markov Logic

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

Weight Function

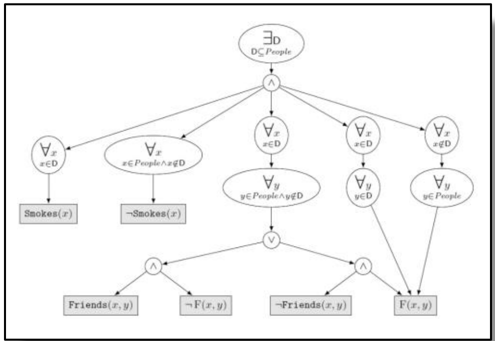
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=exp(3.14)
w(¬F)=1

FOL Sentence

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

Compile?

First-Order d-DNNF Circuit



[Vdb’11,’13]

First-Order Knowledge Compilation

Markov Logic

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

Weight Function

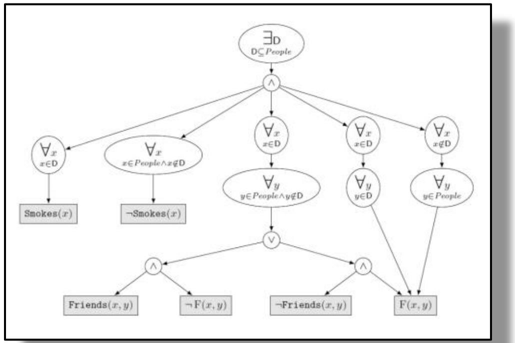
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=exp(3.14)
w(¬F)=1

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

[Vdb'11,'13]

First-Order Knowledge Compilation

Markov Logic

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

Weight Function

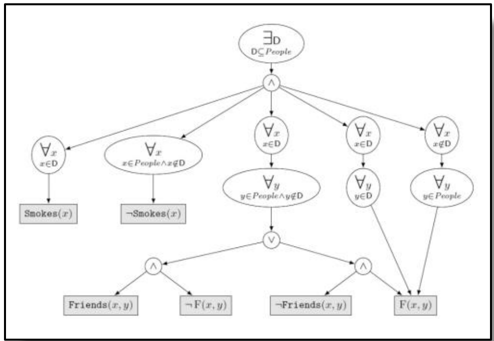
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=exp(3.14)
w(¬F)=1

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

[Vdb'11,'13]

First-Order Knowledge Compilation

Markov Logic

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

Weight Function

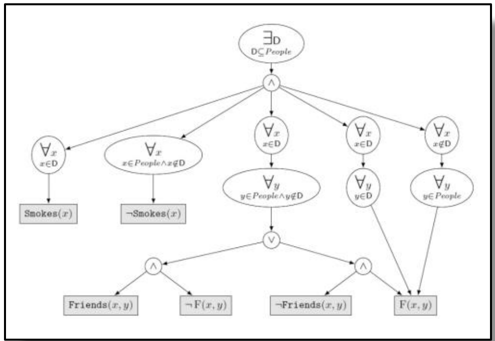
w(Smokes)=1
w(¬Smokes)=1
w(Friends)=1
w(¬Friends)=1
w(F)=exp(3.14)
w(¬F)=1

FOL Sentence

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

Compile?

First-Order d-DNNF Circuit



Domain

Alice
Bob
Charlie

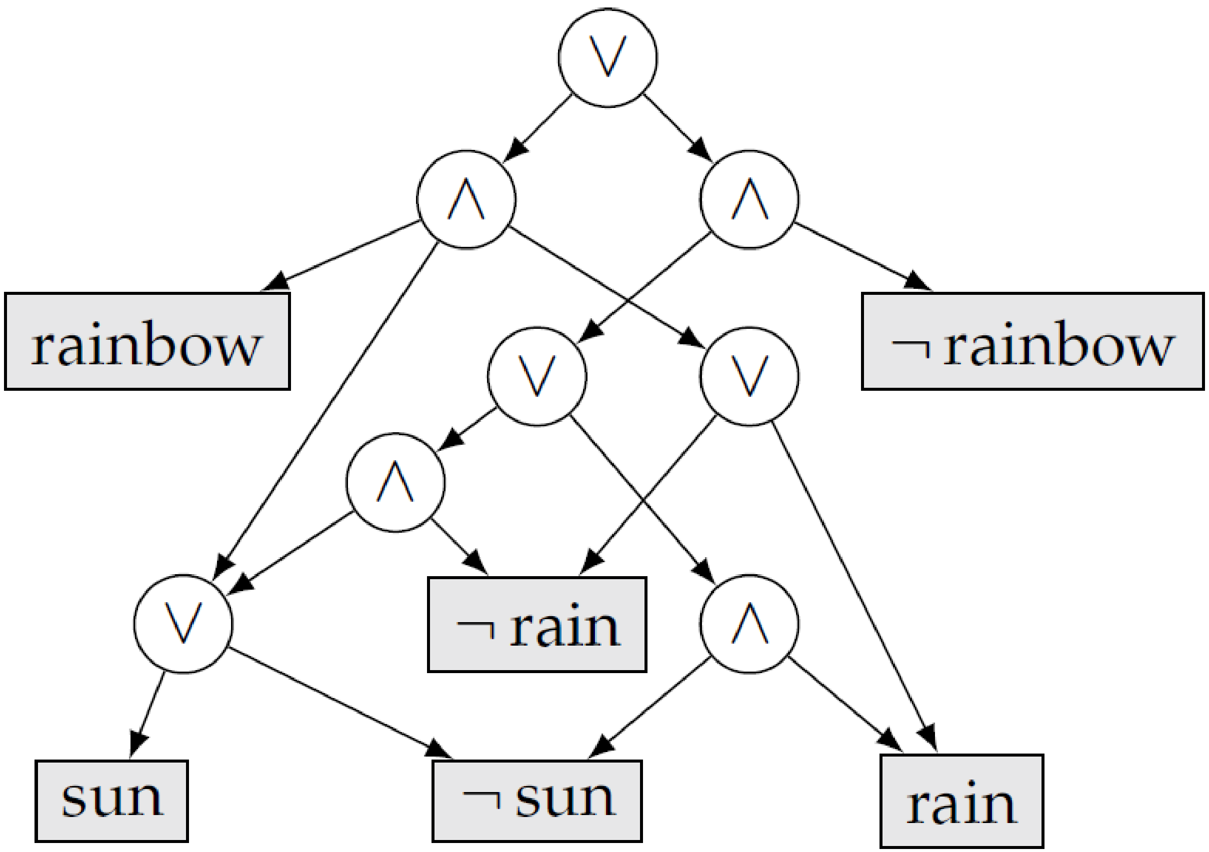
Z = WFOMC = 1479.85

Evaluation in time polynomial in domain size

Domain-lifted!

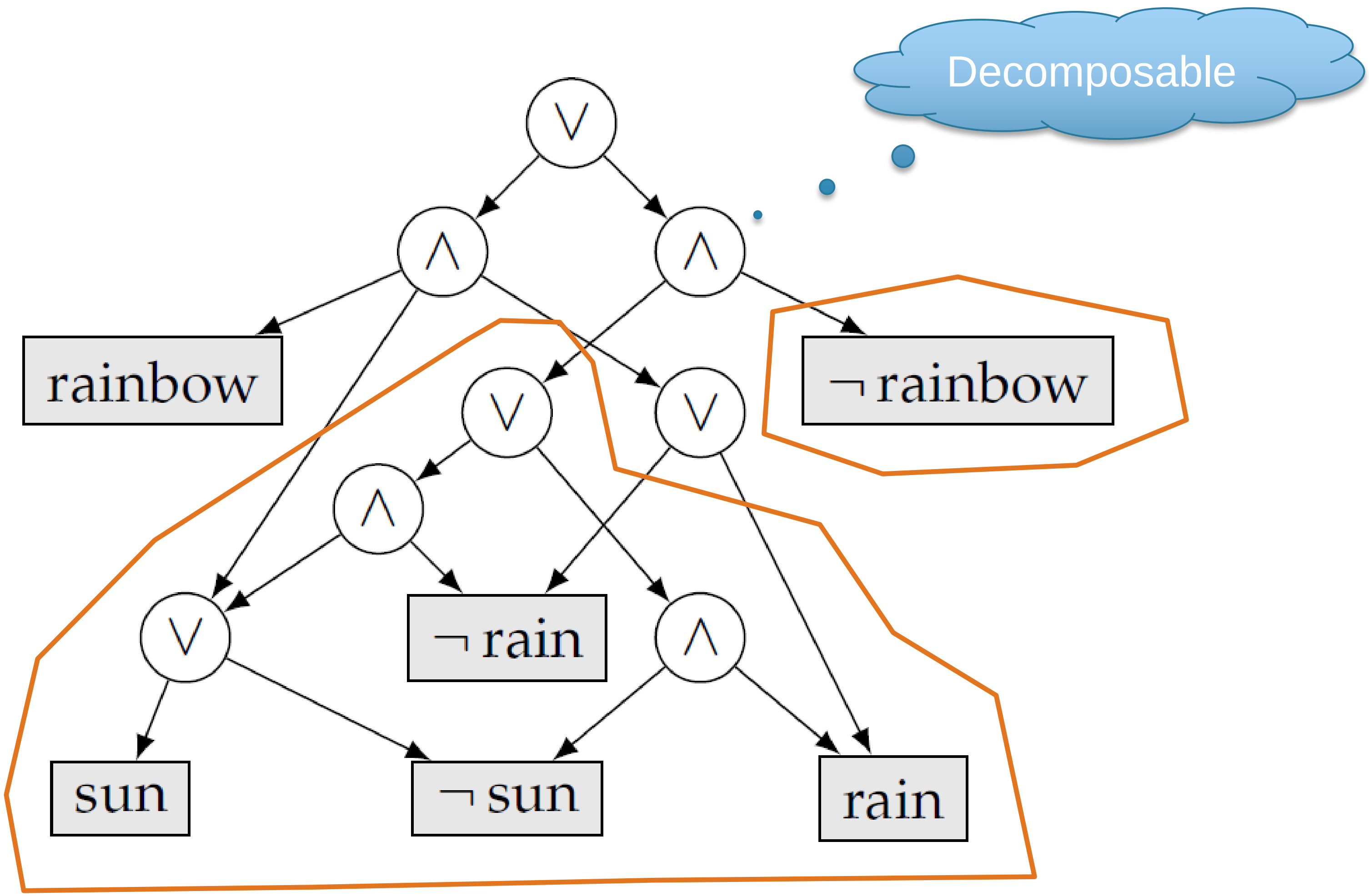
[Vdb'11,'13]

Negation Normal Form



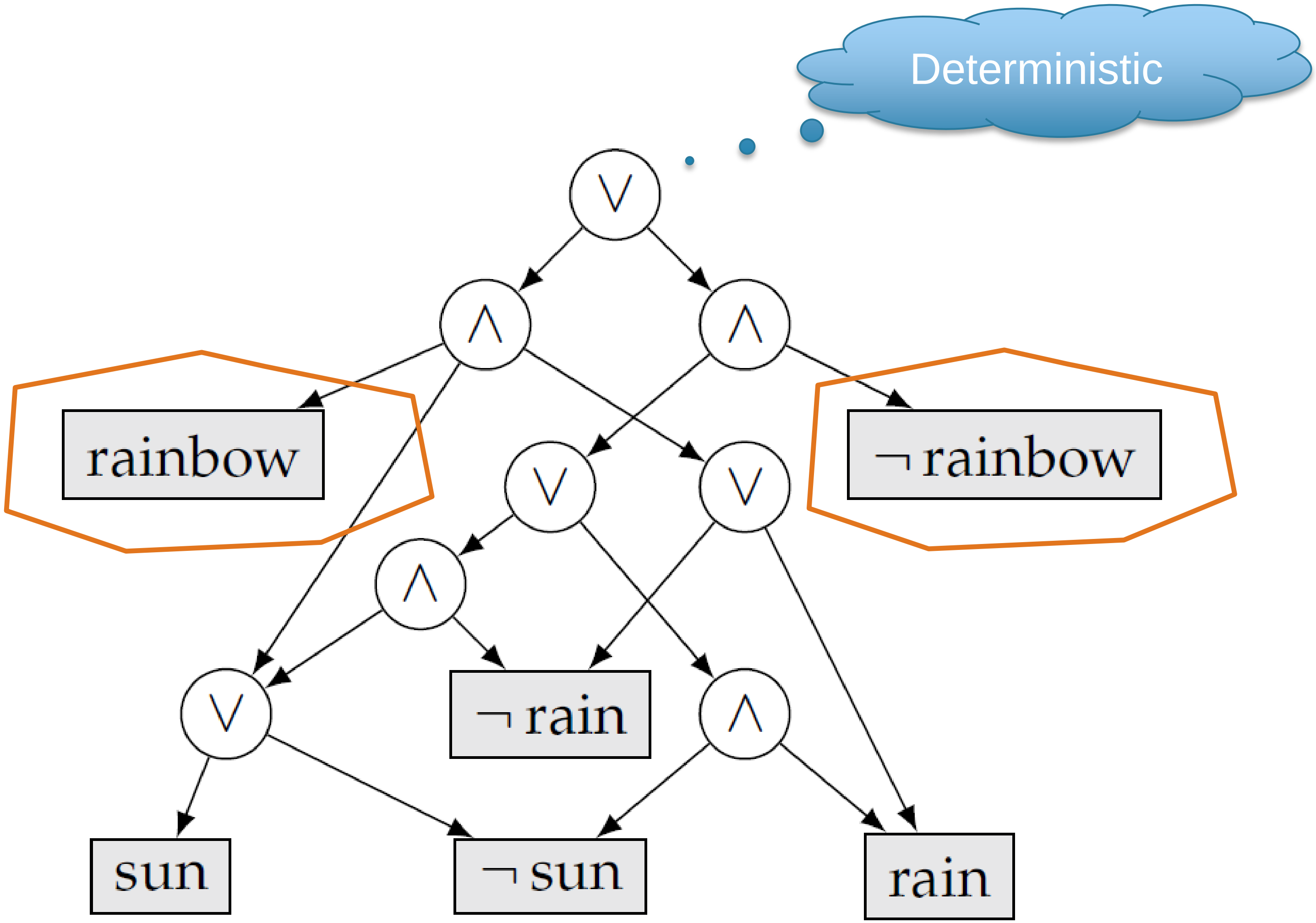
[Darwiche'01]

Decomposable NNF



[Darwiche'01]

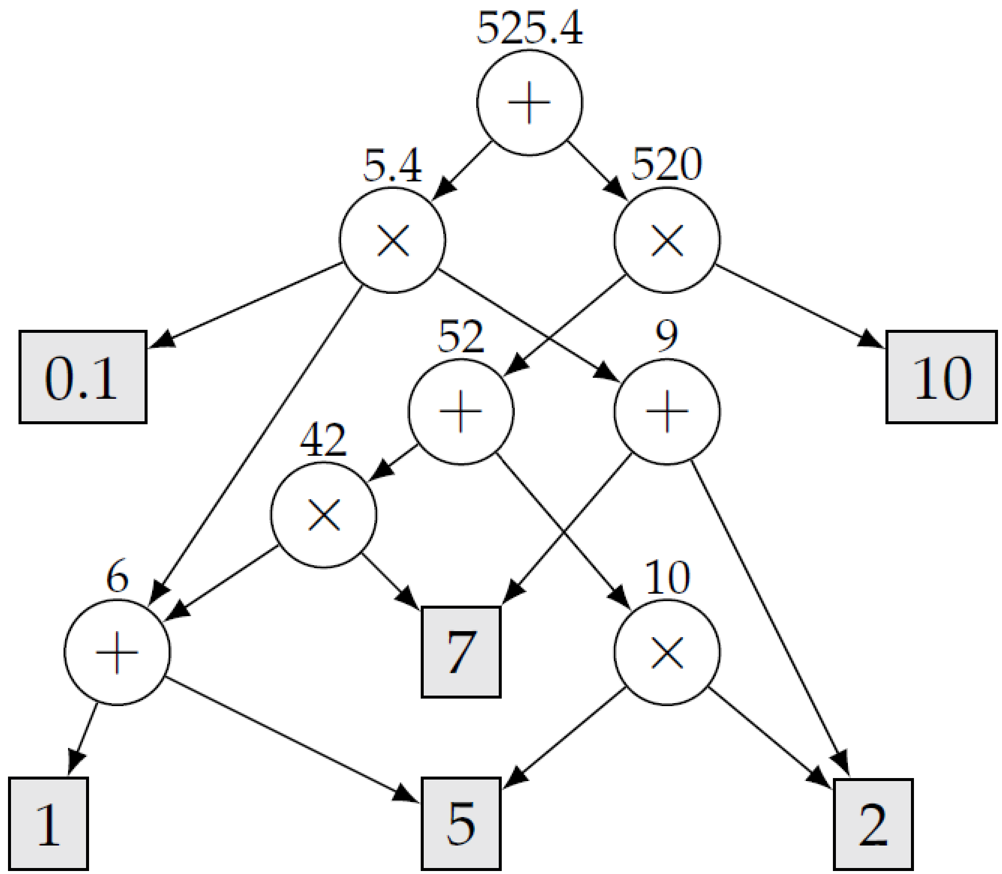
Deterministic Decomposable NNF



[Darwiche'01]

Deterministic Decomposable NNF

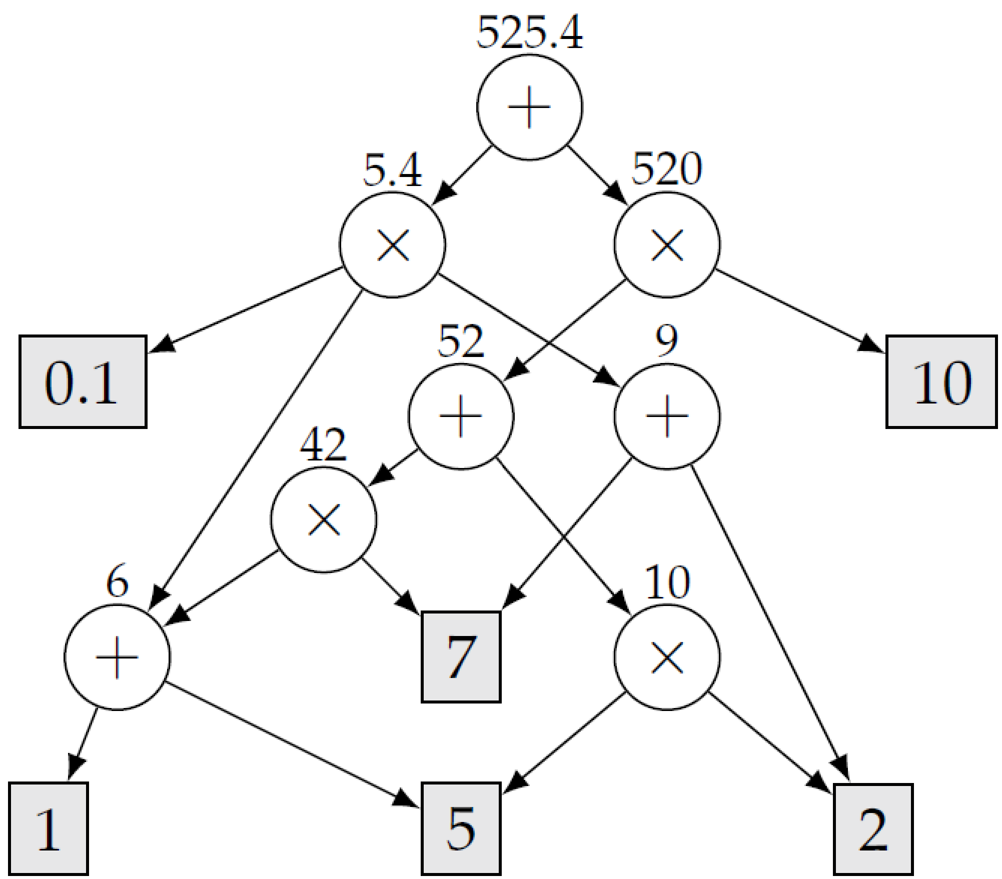
Weighted Model Counting



[Darwiche'01]

Deterministic Decomposable NNF

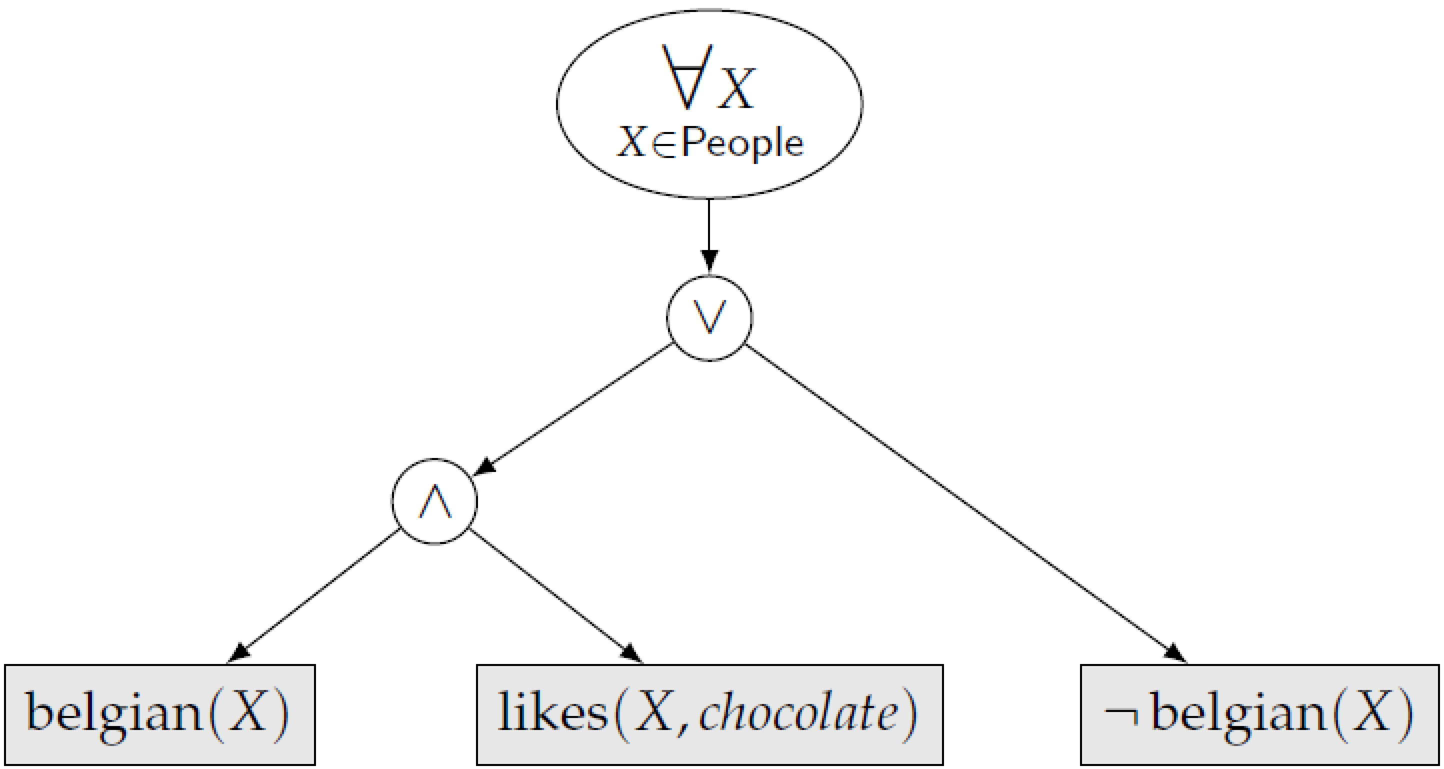
Weighted Model Counting **and much more!**



[Darwiche'01]

First-Order NNF

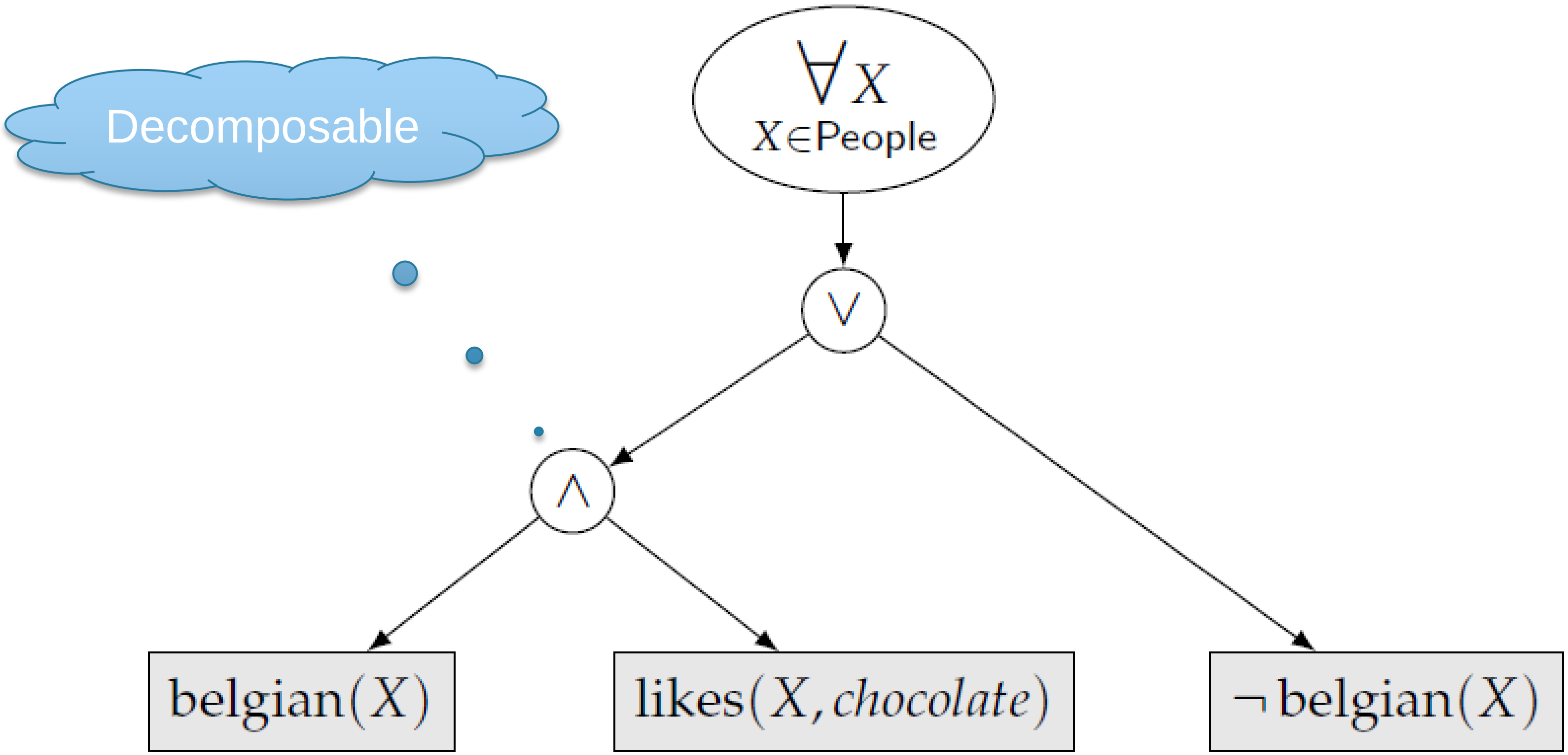
$$\forall X, X \in \text{People} : \text{belgian}(X) \Rightarrow \text{likes}(X, \mathit{chocolate})$$



[VdB'13]

First-Order Decomposability

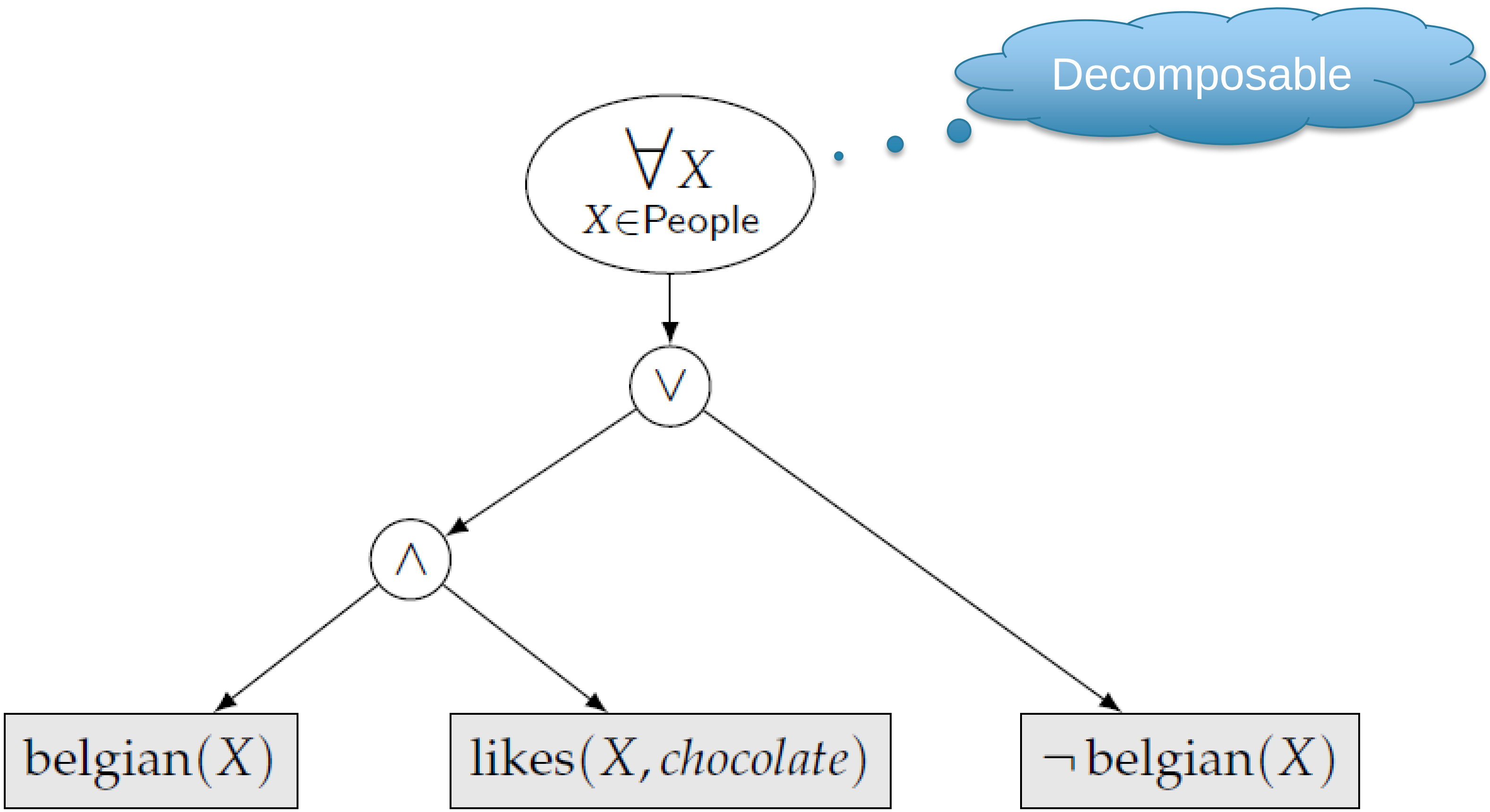
$$\forall X, X \in \text{People} : \text{belgian}(X) \Rightarrow \text{likes}(X, chocolate)$$



[VdB'13]

First-Order Decomposability

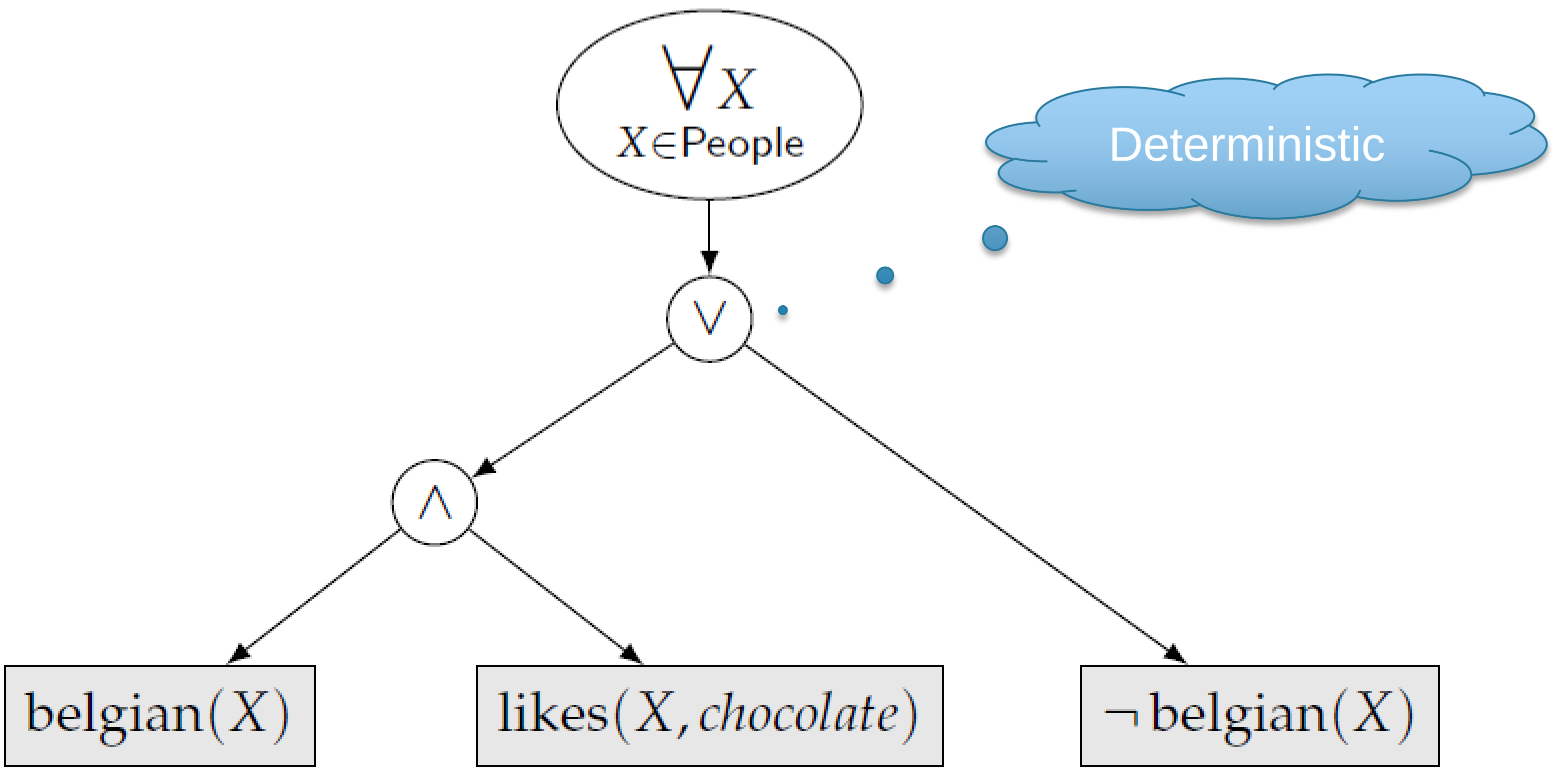
$$\forall X, X \in \mathsf{People} : \mathrm{belgian}(X) \Rightarrow \mathrm{likes}(X, \mathit{chocolate})$$



[VdB'13]

First-Order Determinism

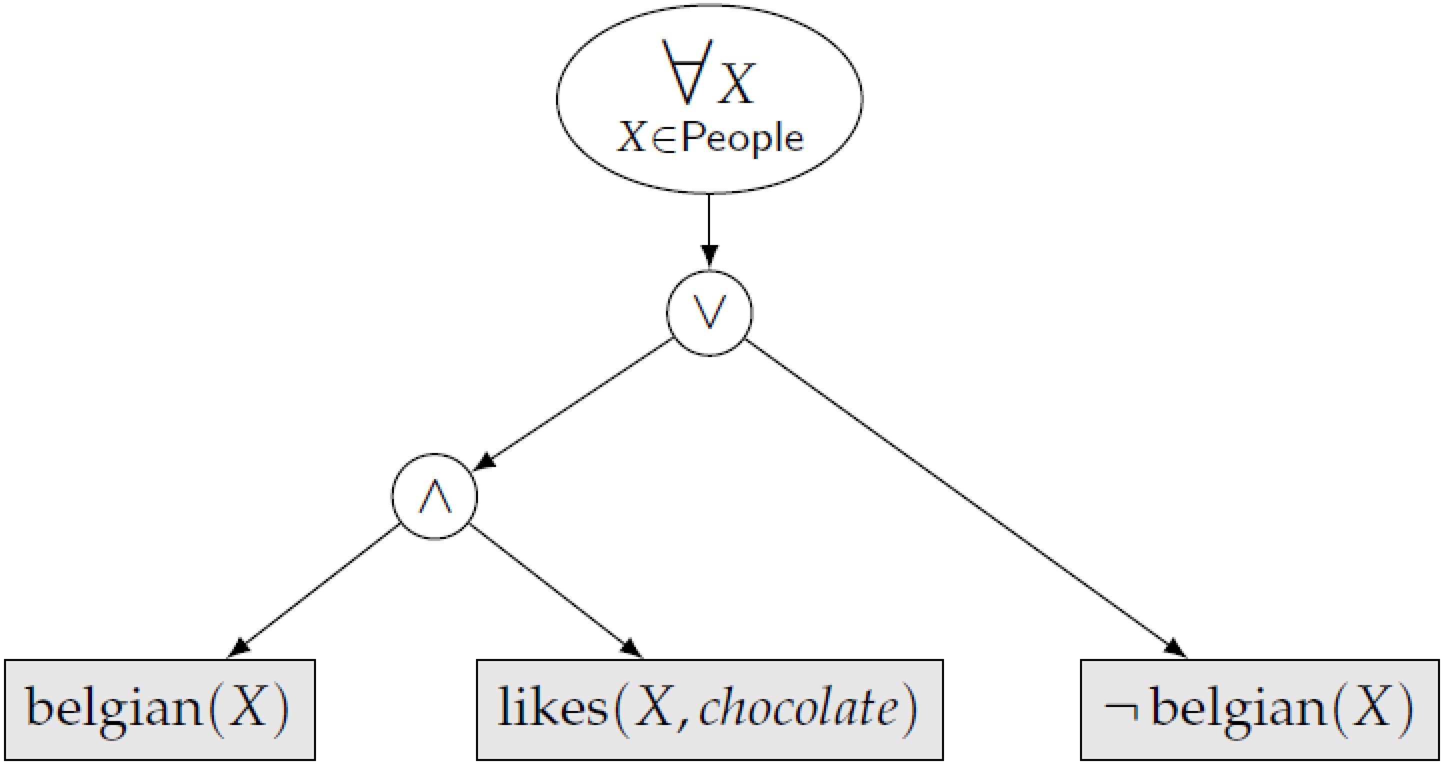
$$\forall X, X \in \text{People} : \text{belgian}(X) \Rightarrow \text{likes}(X, \mathit{chocolate})$$



[VdB'13]

First-Order NNF = Query Plan

$$\forall X, X \in \mathsf{People} : \mathrm{belgian}(X) \Rightarrow \mathrm{likes}(X, \mathit{chocolate})$$

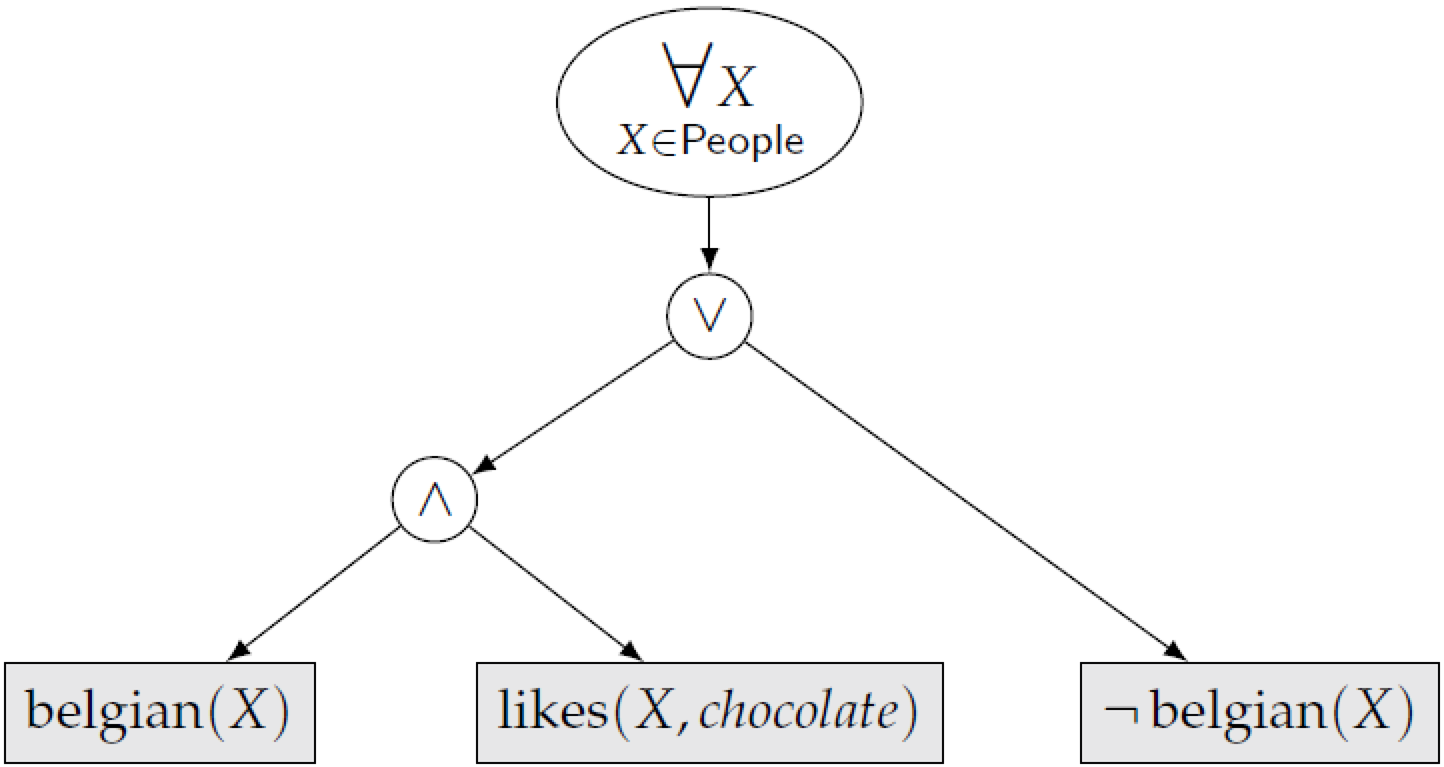


[VdB’13]

Deterministic Decomposable FO NNF

$$\forall X, X \in \mathsf{People} : \mathrm{belgian}(X) \Rightarrow \mathrm{likes}(X, \mathit{chocolate})$$

Weighted Model Counting

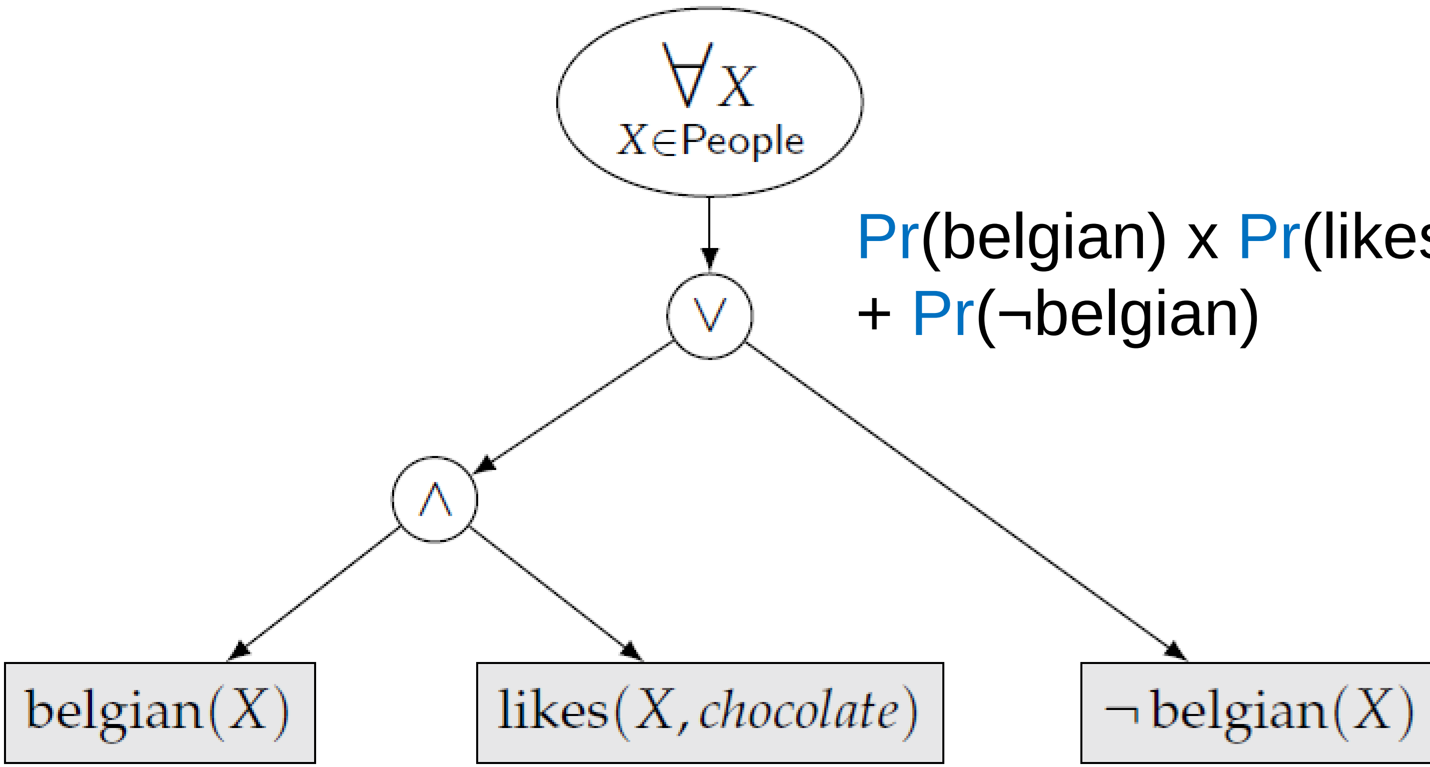


[VdB'13]

Deterministic Decomposable FO NNF

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

Weighted Model Counting



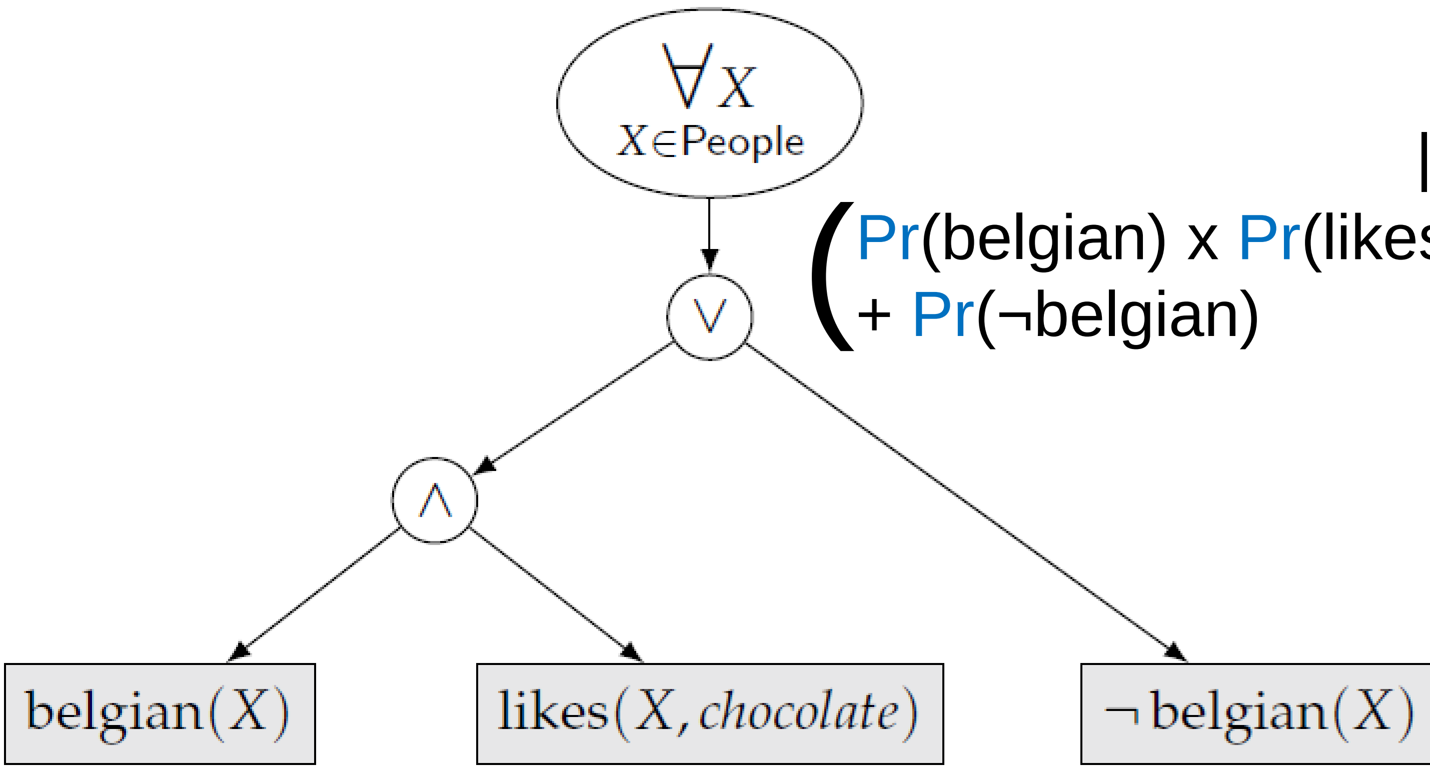
Pr(belgian) x Pr(likes)
+ Pr(¬belgian)

[VdB'13]

Deterministic Decomposable FO NNF

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

Weighted Model Counting



$$\left(\text{Pr(belgian) x Pr(likes)} + \text{Pr}(\neg\text{belgian})\right)^{|\text{People}|}$$

[VdB'13]

Symmetric WFOMC on FO NNF

$$U(\alpha) = \begin{cases} 0 & \textit{when } \alpha = \mathsf{false} \\ 1 & \textit{when } \alpha = \mathsf{true} \\ 0.5 & \textit{when } \alpha \textit{ is a literal} \\ U(\ell_1) \times \cdots \times U(\ell_n) & \textit{when } \alpha = \ell_1 \wedge \cdots \wedge \ell_n \\ U(\ell_1) + \cdots + U(\ell_n) & \textit{when } \alpha = \ell_1 \vee \cdots \vee \ell_n \\ \prod_{i=1}^{n} U(\beta\{X/x_i\}) & \textit{when } \alpha = \forall X \in \tau, \beta \textit{ and } x_1, \ldots, x_n \textit{ are the objects in } \tau. \\ \sum_{i=1}^{n} U(\beta\{X/x_i\}) & \textit{when } \alpha = \exists X \in \tau, \beta \textit{ and } x_1, \ldots, x_n \textit{ are the objects in } \tau. \\ \prod_{i=0}^{|\tau|} U(\beta\{\mathbf{X}/\mathbf{x}_i\})^{\binom{|\tau|}{i}} & \textit{when } \alpha = \forall \mathbf{X} \subseteq \tau, \beta, \textit{ and } \mathbf{x}_i \textit{ is any subset of } \tau \textit{ such that } |\mathbf{x}_i| = i. \\ \sum_{i=0}^{|\tau|} \binom{|\tau|}{i} \cdot U(\beta\{\mathbf{X}/\mathbf{x}_i\}) & \textit{when } \alpha = \exists \mathbf{X} \subseteq \tau, \beta, \textit{ and } \mathbf{x}_i \textit{ is any subset of } \tau \textit{ such that } |\mathbf{x}_i| = i. \end{cases}$$

Complexity polynomial in domain size!
Polynomial in NNF size for bounded depth.

[VdB'13]

How to do first-order knowledge ***compilation****?*

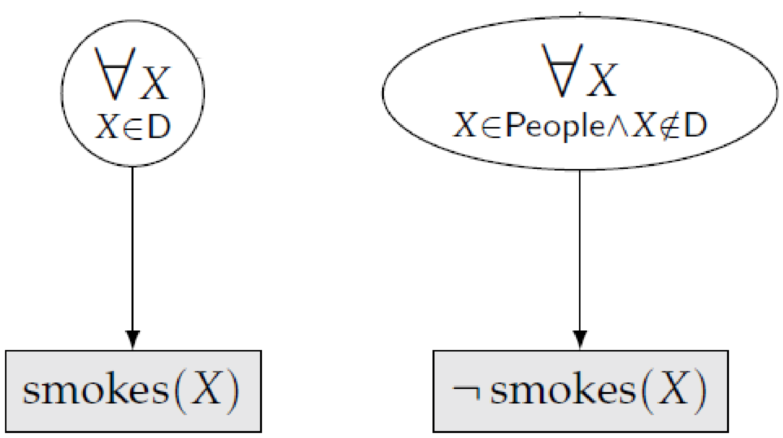
Deterministic Decomposable FO NNF

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

[VdB'13]

Deterministic Decomposable FO NNF

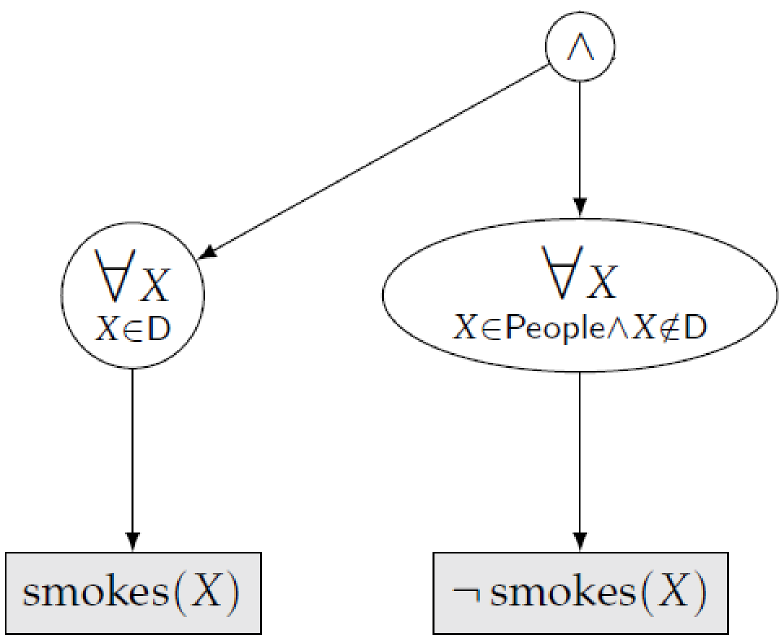
$\Delta = \forall x, y \in \textbf{People}, (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$



[VdB'13]

Deterministic Decomposable FO NNF

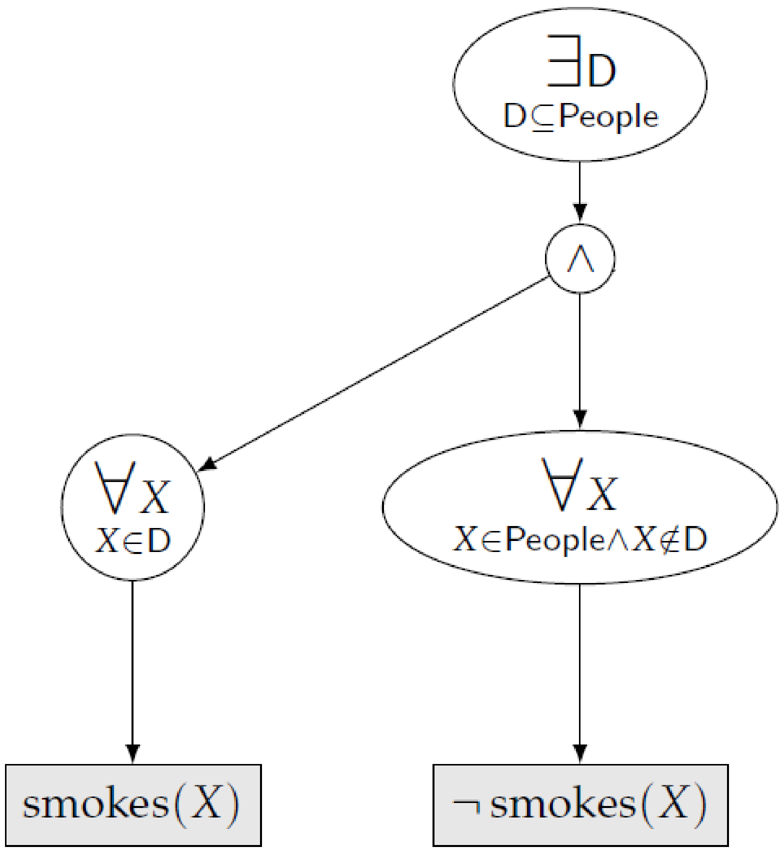
Δ = ∀x ,y ∈ **People**, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))



[VdB'13]

Deterministic Decomposable FO NNF

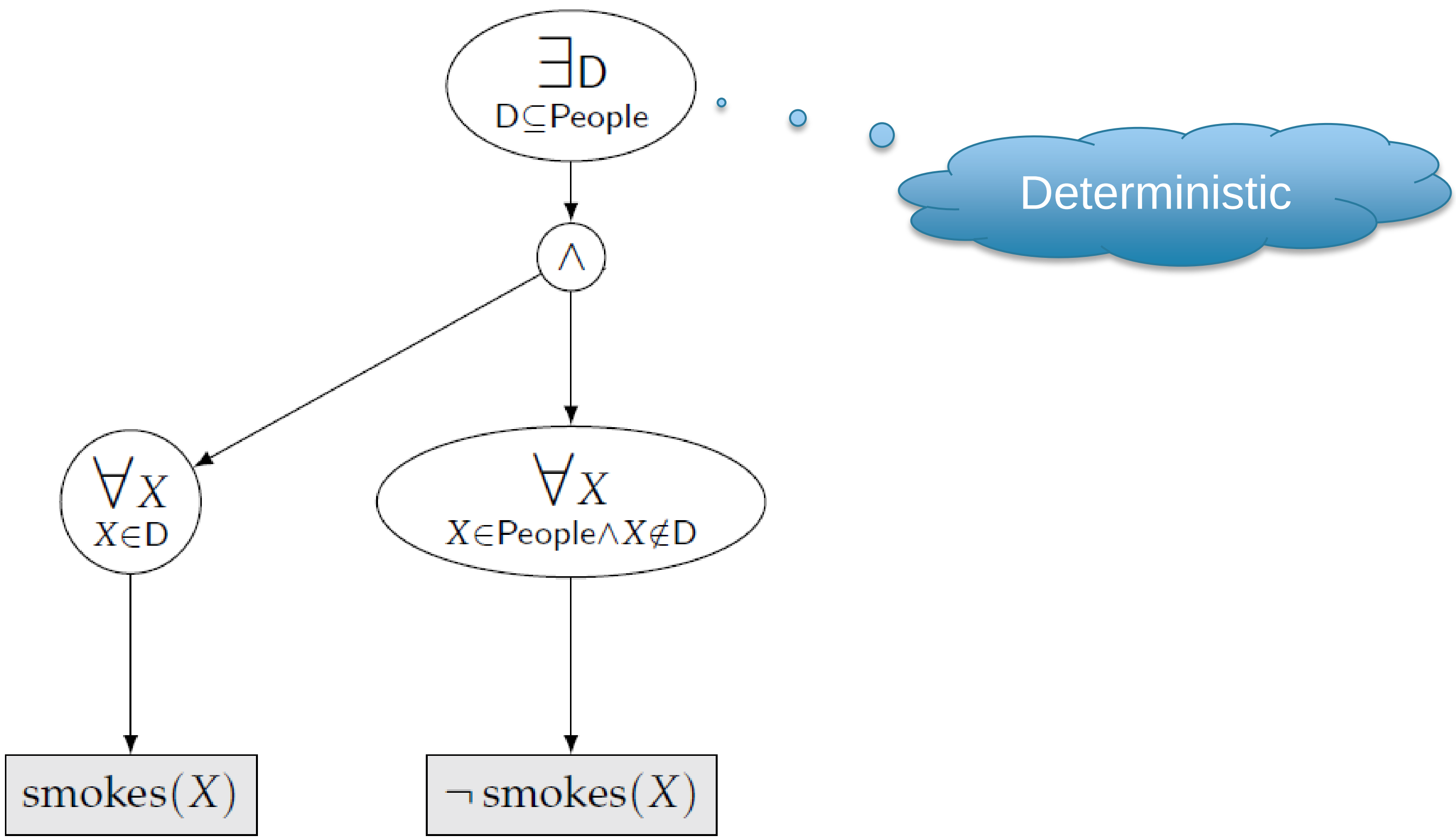
Δ = ∀x ,y ∈ **People**, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))



[VdB'13]

Deterministic Decomposable FO NNF

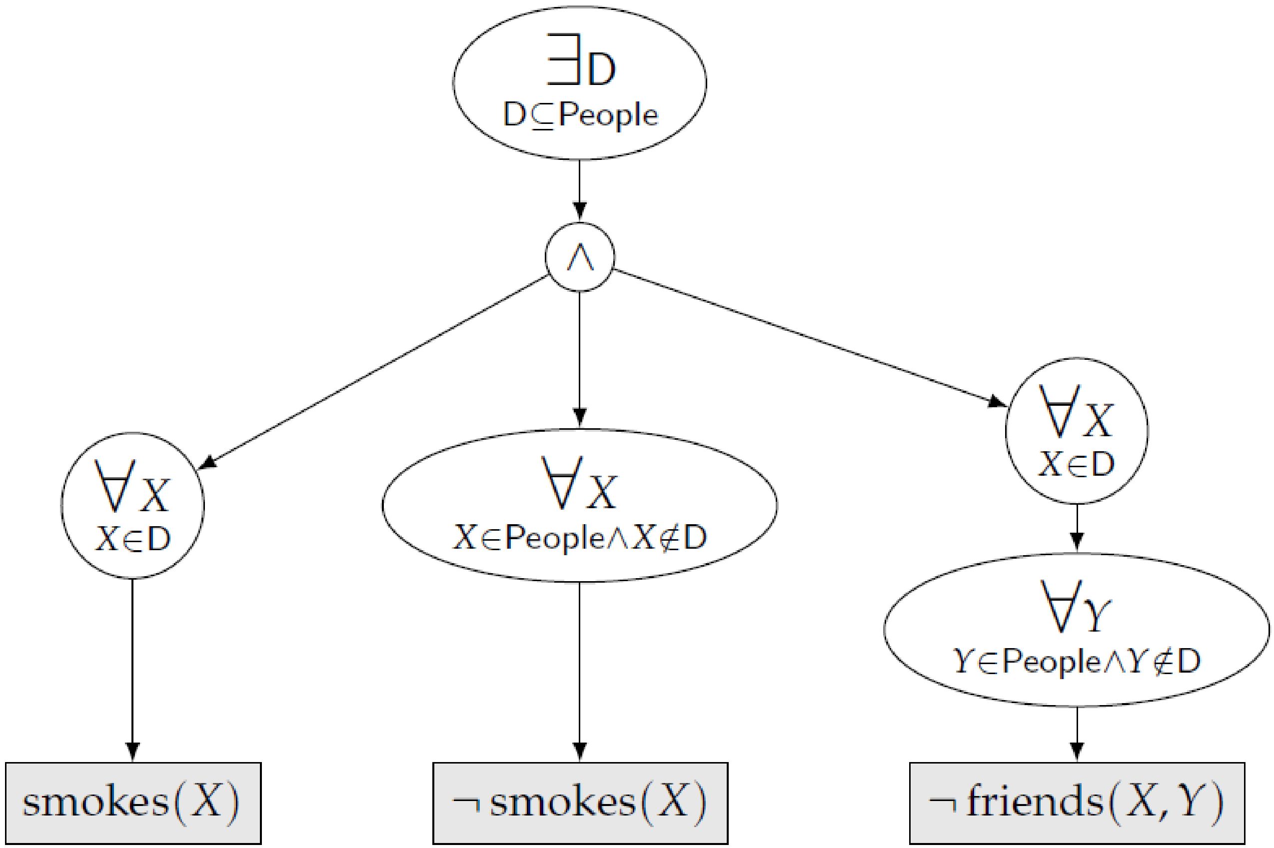
$\Delta = \forall x, y \in$ **People**, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))



[VdB'13]

Deterministic Decomposable FO NNF

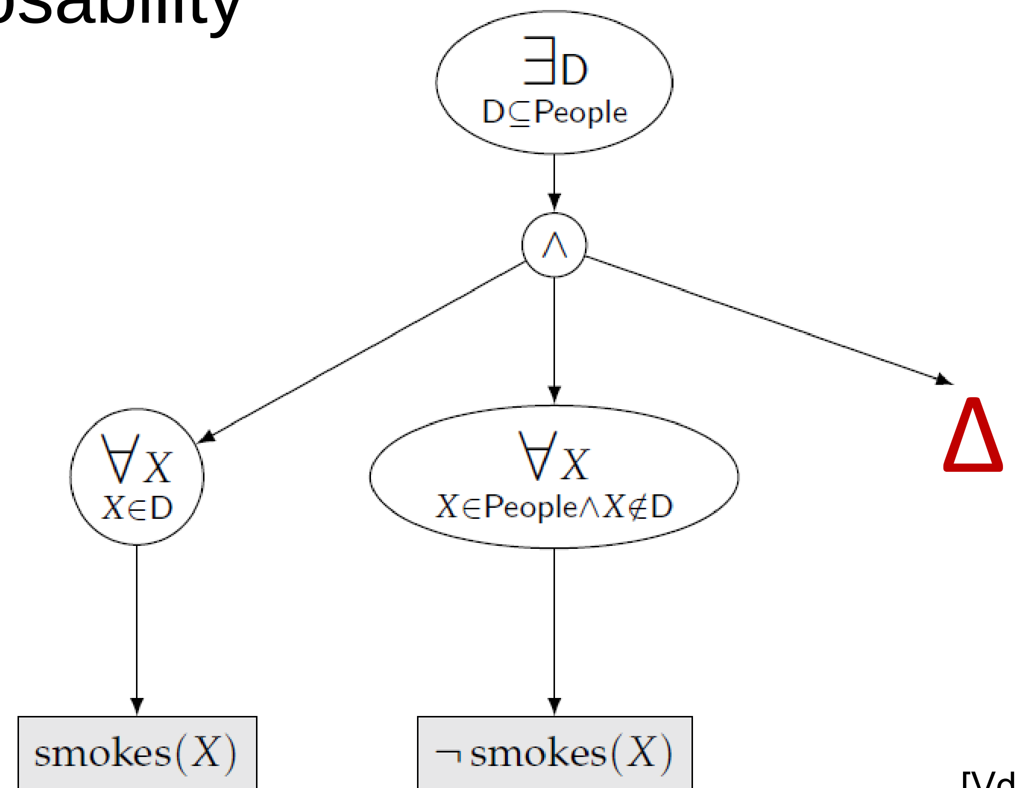
Δ = ∀x ,y ∈ **People**, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))



[VdB'13]

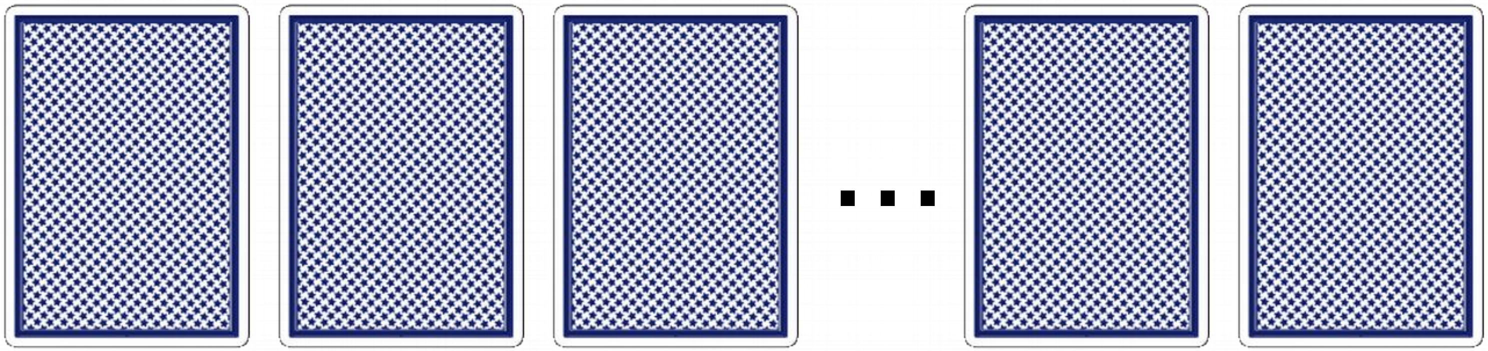
Compilation Rules

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



[VdB'13]

Playing Cards Revisited



Let us automate this:

- **Relational** model

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

- **Lifted** probabilistic inference algorithm

Why not do propositional WMC?

Reduce to propositional model counting:

[VdB'15]

Why not do propositional WMC?

Reduce to propositional model counting:

$$
\begin{aligned}
\Delta = \quad & \text{Card}(A\heartsuit,p_1) \vee \ldots \vee \text{Card}(2\clubsuit,p_1) \\
& \text{Card}(A\heartsuit,p_2) \vee \ldots \vee \text{Card}(2\clubsuit,p_2) \\
& \ldots \\
& \text{Card}(A\heartsuit,p_1) \vee \ldots \vee \text{Card}(A\heartsuit,p_{52}) \\
& \text{Card}(K\heartsuit,p_1) \vee \ldots \vee \text{Card}(K\heartsuit,p_{52}) \\
& \ldots \\
& \neg\text{Card}(A\heartsuit,p_1) \vee \neg\text{Card}(A\heartsuit,p_2) \\
& \neg\text{Card}(A\heartsuit,p_1) \vee \neg\text{Card}(A\heartsuit,p_3) \\
& \ldots
\end{aligned}
$$

[VdB'15]

Why not do propositional WMC?

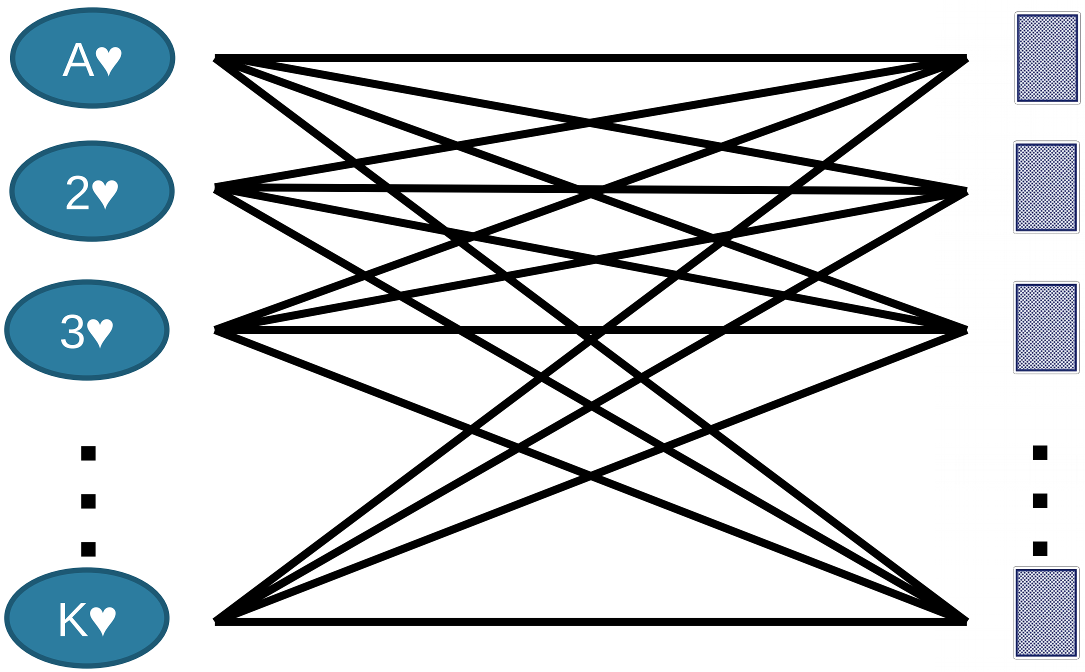
Reduce to propositional model counting:

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

What will happen?

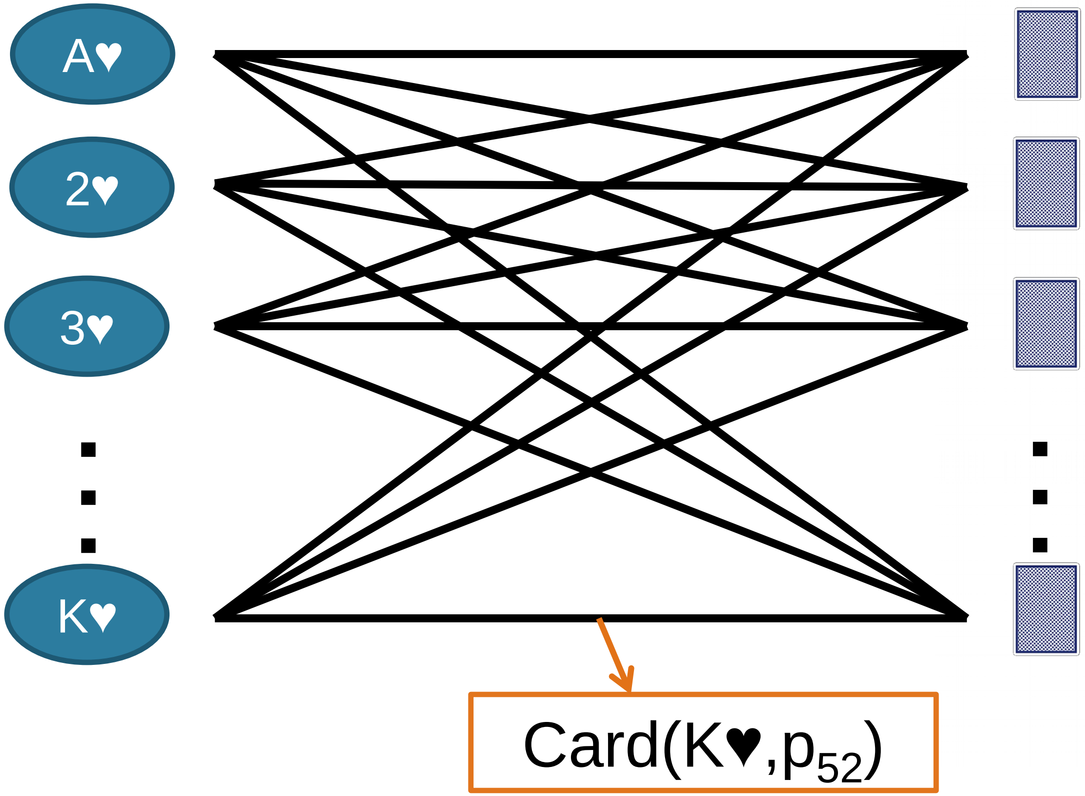
[VdB'15]

Deck of Cards Graphically



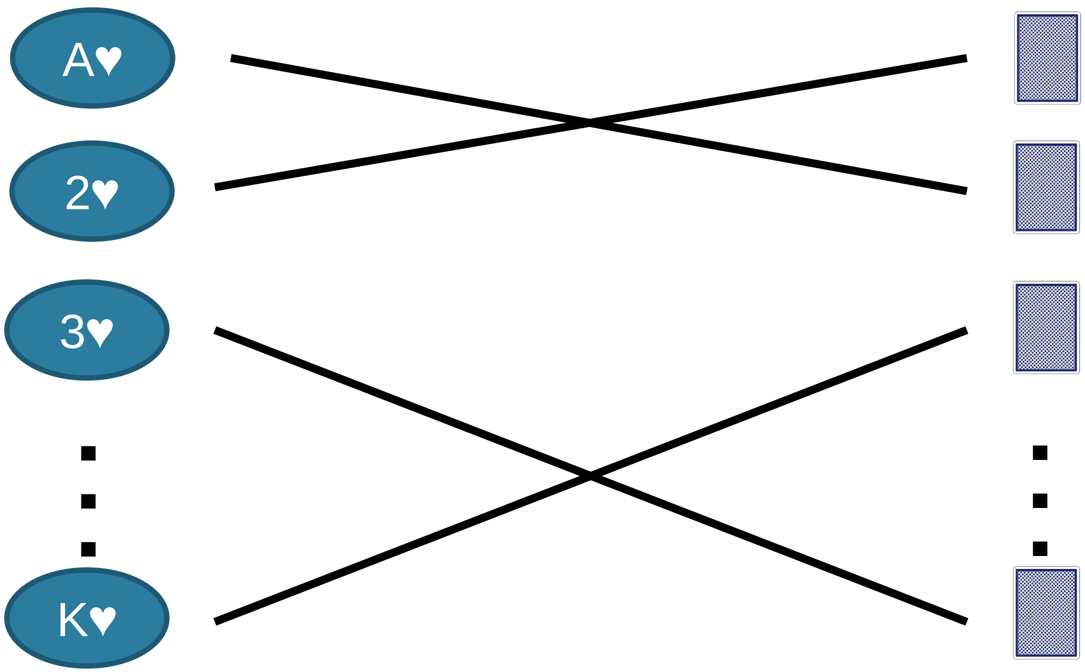
[VdB'15]

Deck of Cards Graphically



[VdB'15]

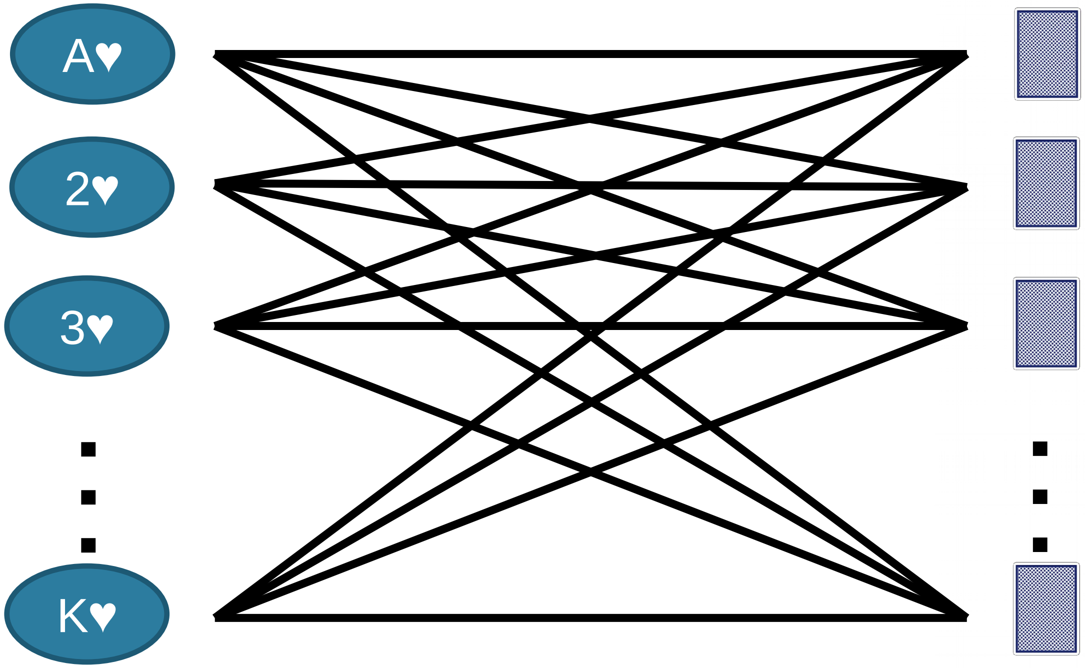
Deck of Cards Graphically



One model/*perfect matching*

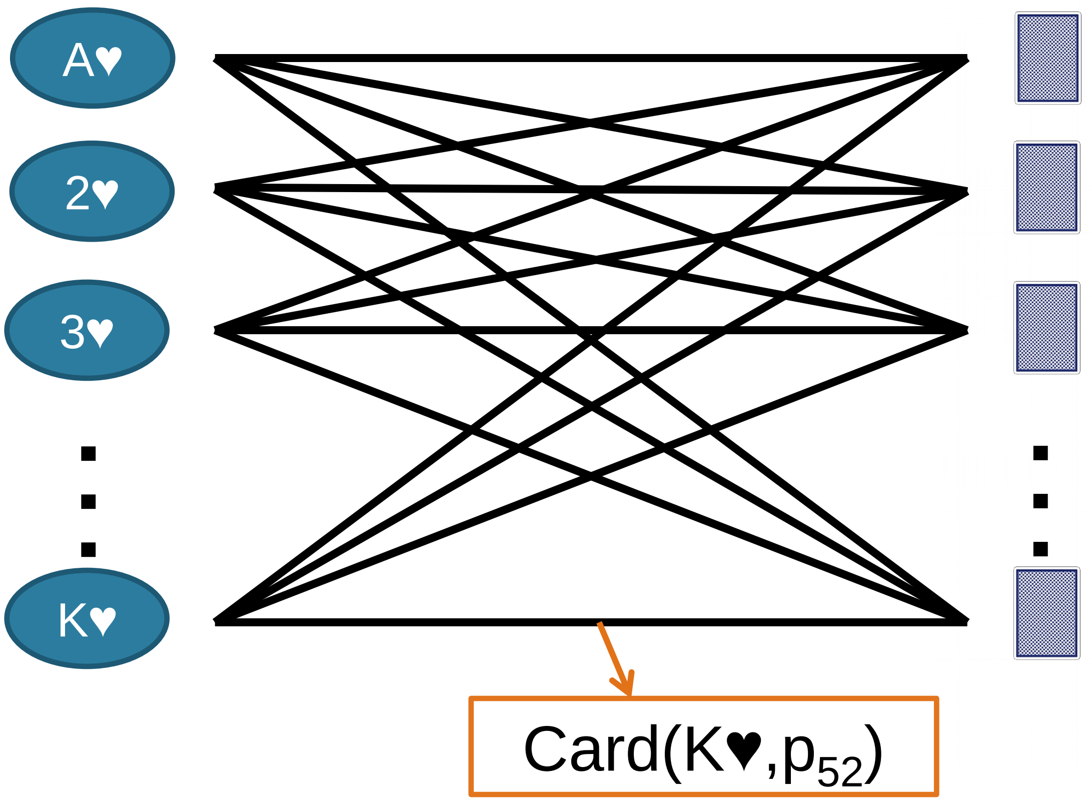
[VdB'15]

Deck of Cards Graphically



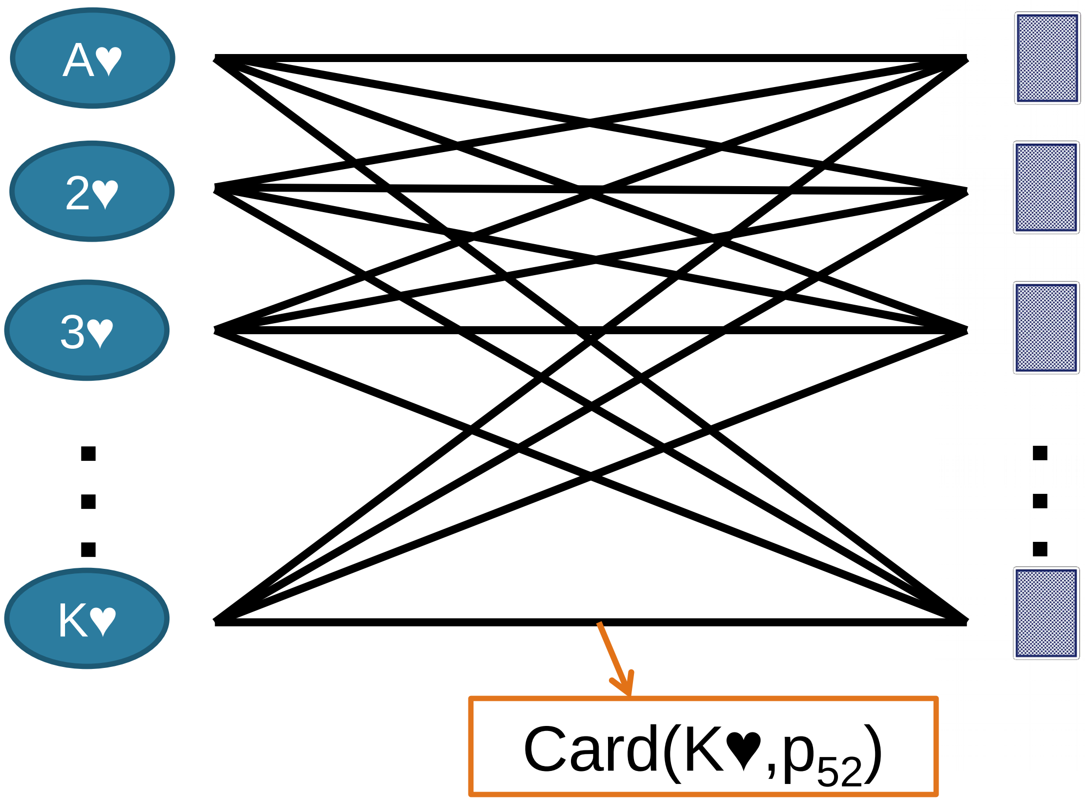
[VdB'15]

Deck of Cards Graphically



[VdB'15]

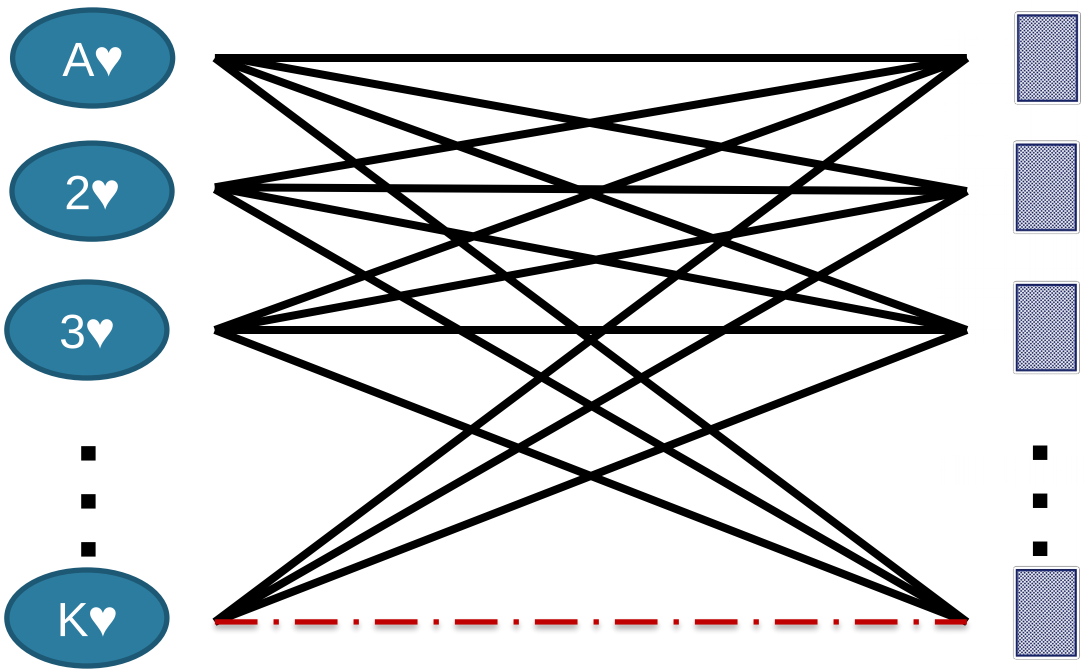
Deck of Cards Graphically



Model counting: How many *perfect matchings*?

[VdB'15]

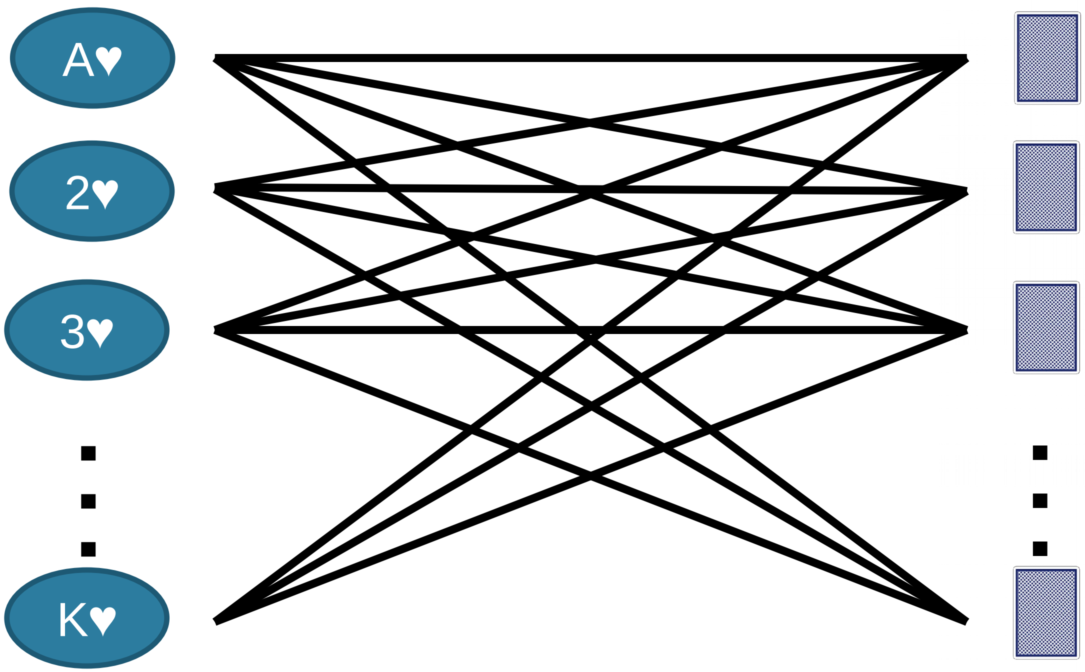
Deck of Cards Graphically



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

[VdB'15]

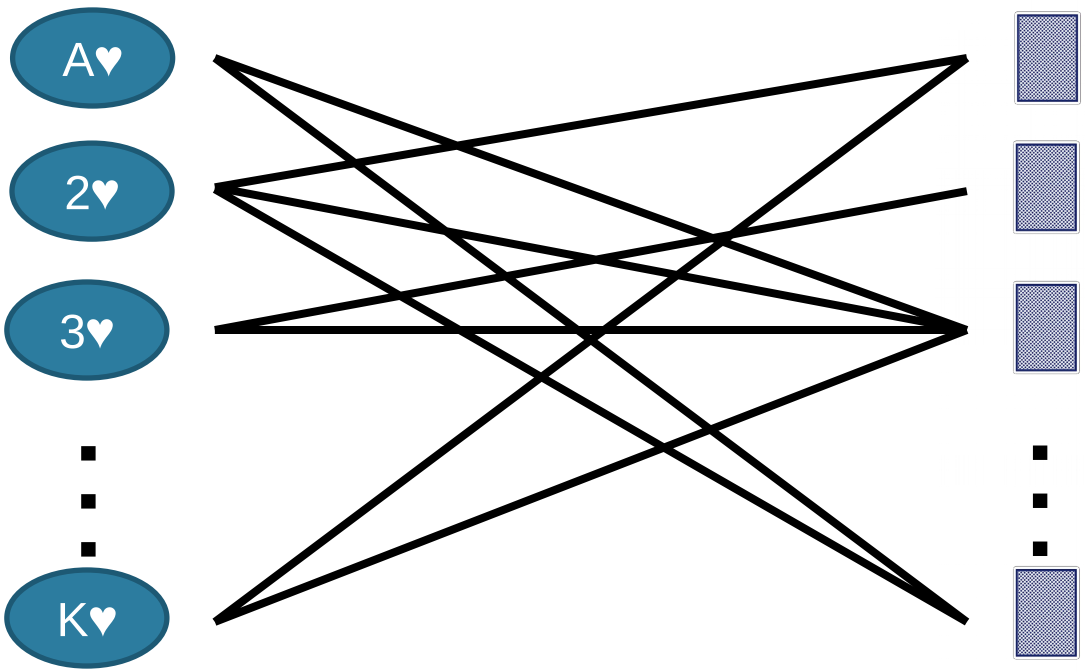
Deck of Cards Graphically



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

[VdB'15]

Deck of Cards Graphically



What if I set can set any
asymmetric weight function?

[VdB'15]

Observations

- Asymmetric weight function can remove edge
  Encode any bigraph

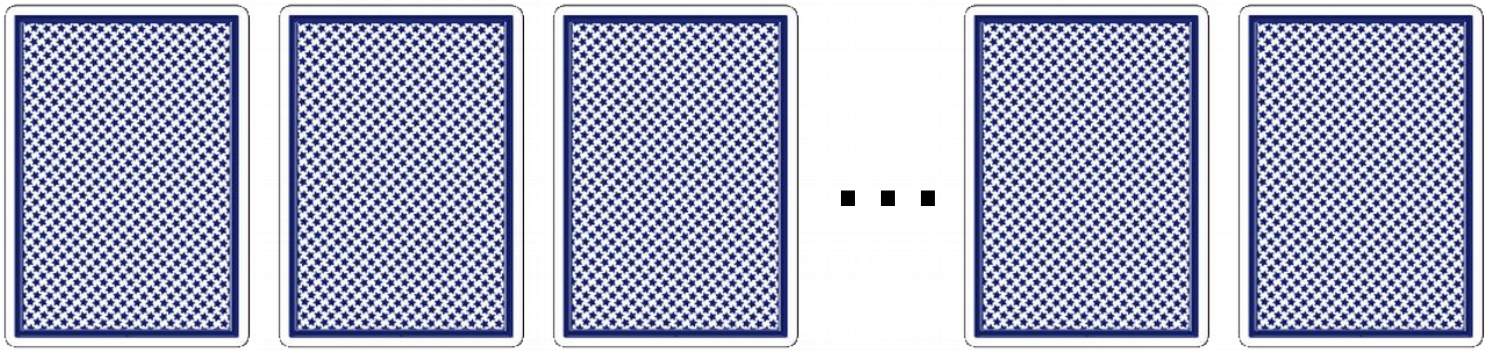
- Counting models = perfect matchings
- Problem is #P-complete! ☹

- All non-lifted WMC solvers efficiently handle asymmetric weights
- No solver does cards problem efficiently!

Later: Power of lifted vs. ground inference and complexities

[VdB'15]

Playing Cards Revisited



Let us automate this:

- **Relational** model

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

- **Lifted** probabilistic inference algorithm

Playing Cards Revisited

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

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

$$\forall p, \forall c, \mathrm{Card}(p,c) \Rightarrow S_1(p)$$
$$\forall c, \forall p, \mathrm{Card}(p,c) \Rightarrow S_2(c)$$
$$\forall p, \forall c, \forall c', \mathrm{Card}(p,c) \land \mathrm{Card}(p,c') \Rightarrow c = c'$$

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

$$\forall p, \forall c, Card(p,c) \Rightarrow S_1(p)$$
$$\forall c, \forall p, Card(p,c) \Rightarrow S_2(c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

[VdB'15]

Playing Cards Revisited

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

↓ Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

↓ Atom counting

[VdB'15]

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

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

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

$\forall p, \forall c, Card(p,c) \Rightarrow \cancel{S_1(p)}$
$\forall c, \forall p, Card(p,c) \Rightarrow \cancel{S_2(c)}$
$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

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

$\downarrow$ $\forall$-Rule

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

$$\forall p, \forall c, Card(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, Card(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', Card(p,c) \land Card(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

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

$\downarrow$ $\forall$-Rule

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

[VdB'15]

Playing Cards Revisited

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

$\downarrow$ Skolemization

$$\forall p, \forall c, \mathrm{Card}(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, \mathrm{Card}(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', \mathrm{Card}(p,c) \land \mathrm{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

$$\forall p, \forall c, \forall c', \mathrm{Card}(p,c) \land \mathrm{Card}(p,c') \Rightarrow c = c'$$

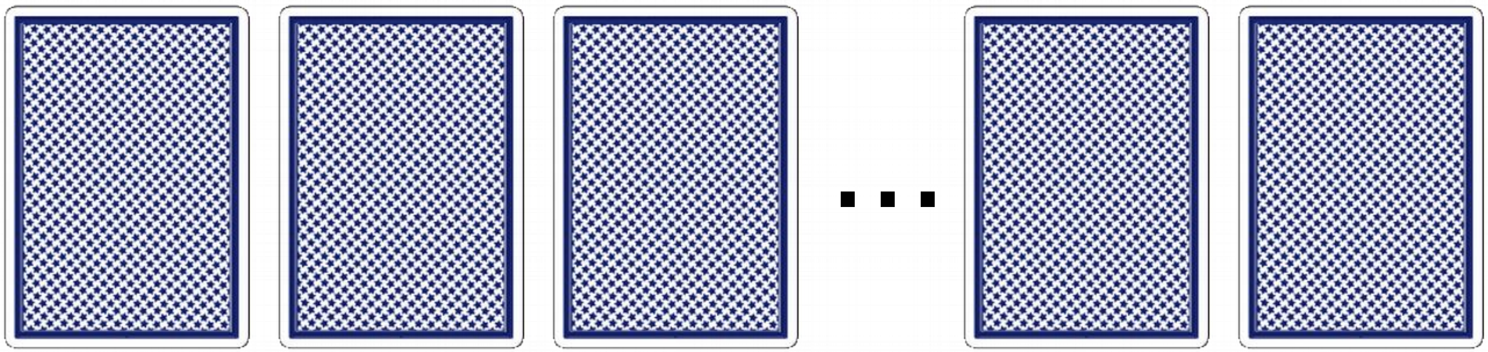
$\downarrow$ $\forall$-Rule

$$\forall c, \forall c', \mathrm{Card}(c) \land \mathrm{Card}(c') \Rightarrow c = c'$$

$\downarrow$ …

[VdB'15]

Playing Cards Revisited



Let us automate this:

– **Lifted** probabilistic inference algorithm

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

Computed in time polynomial in n

[VdB'15]

Summary Lifted Inference

- By definition: PTIME data complexity
  Also: ∃ FO compilation = ∃ Query Plan
- However: only works for "liftable" queries
- Preprocessing based on logical rewriting
- The rules: Deceptively simple: the only surprising rules are I/E and atom counting
- Rules are captured by a query plan
  or first-order NNF circuit