Intelligent Agents:
Web-mining Agents

Probabilistic Graphical Models

Lifted Inference

Tanya Braun

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Probabilistic Graphical Models (PGMs)

1. Recap: **Propositional** modelling
   - Factor model, Bayesian network, Markov network
   - Semantics, inference tasks + algorithms + complexity
2. **Probabilistic relational models** (PRMs)
   - Parameterised models, Markov logic networks
   - Semantics, inference tasks
3. **Lifted inference**
   - LVE, LJT, FOKC
   - Theoretical analysis
4. **Lifted learning**
   - Recap: propositional learning
   - From ground to lifted models
   - Direct lifted learning
5. **Approximate Inference: Sampling**
   - Importance sampling
   - MCMC methods
6. **Sequential models & inference**
   - Dynamic PRMs
   - Semantics, inference tasks + algorithms + complexity, learning
7. **Decision making**
   - (Dynamic) Decision PRMs
   - Semantics, inference tasks + algorithms, learning
8. **Continuous Space**
   - Gaussian distributions and Bayesian networks
   - Probabilistic soft logic

Lifting? → Inverse of grounding!

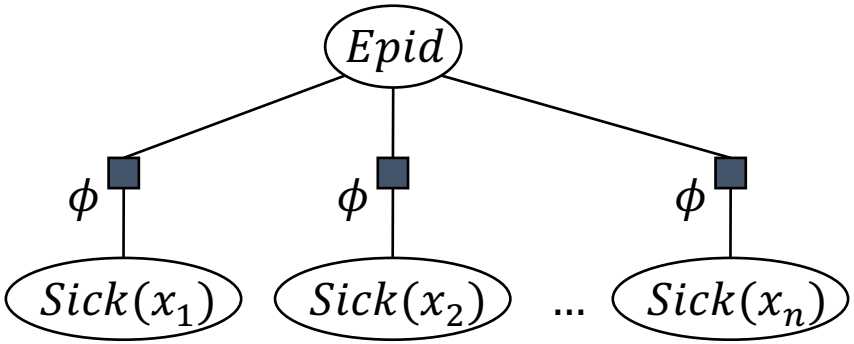
$$P(Epid)$$

$$\propto \sum_{s \in \mathcal{R}(Sick(x_1))} \phi(Epid, Sick(x_1) = s)$$

$$\cdot \sum_{s \in \mathcal{R}(Sick(x_2))} \phi(Epid, Sick(x_2) = s)$$

$$\cdot \dots$$

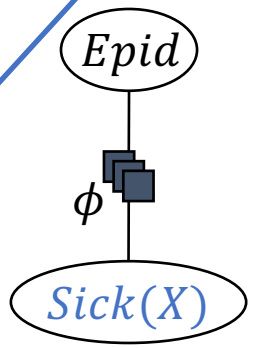
$$\cdot \sum_{s \in \mathcal{R}(Sick(x_n))} \phi(Epid, Sick(x_n) = s)$$



$$= \underbrace{\phi'(Epid) \cdot \phi'(Epid) \cdot \dots \cdot \phi'(Epid)}_{n \text{ times}} = \left(\phi'(Epid)\right)^n$$

of X relative to $\phi'(Epid)$

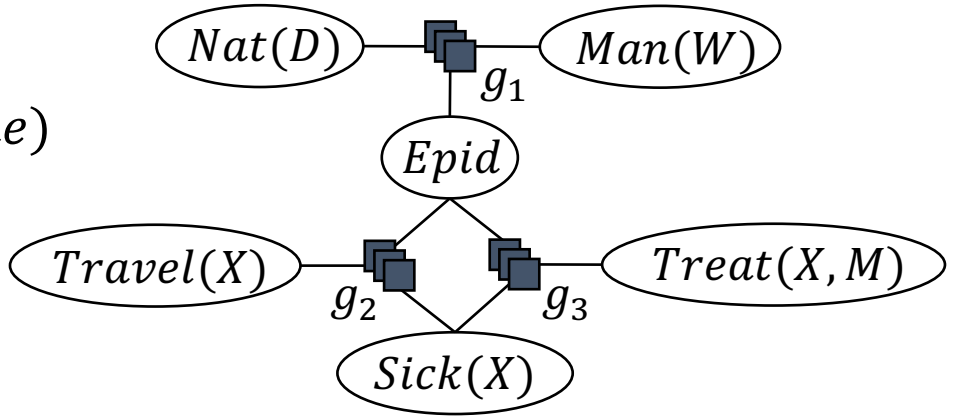
$$P(Epid) \propto \left(\sum_{s \in r(Sick(x))} \phi(Epid, Sick(x) = s) \right)^n = \left(\phi'(Epid)\right)^n$$



Representative for X in $\phi(Epid, Sick(X))$

Queries in Parameterised Models

- Again as before:
- Query Answering Problem
  - Compute an answer to a query $P(\boldsymbol{S}|\boldsymbol{T})$ given a model G representing the full joint probability distribution P_G
    - Query for a marginal (conditional) probability (distribution)
    - Avoid grounding (parts of) G
  - E.g.,
    - $P(Treat(eve, m_1))$
    - $P(Travel(eve), Epid)$
    - $P(Sick(eve)|Epid)$
    - $P(Epid|Sick(eve) = true)$
    - If a CRV occurs:
      $P(\#_E[Epid(E)])$
      $P(\#_E[Epid(E)] = [2,2])$



Semantics of Parameterised Models

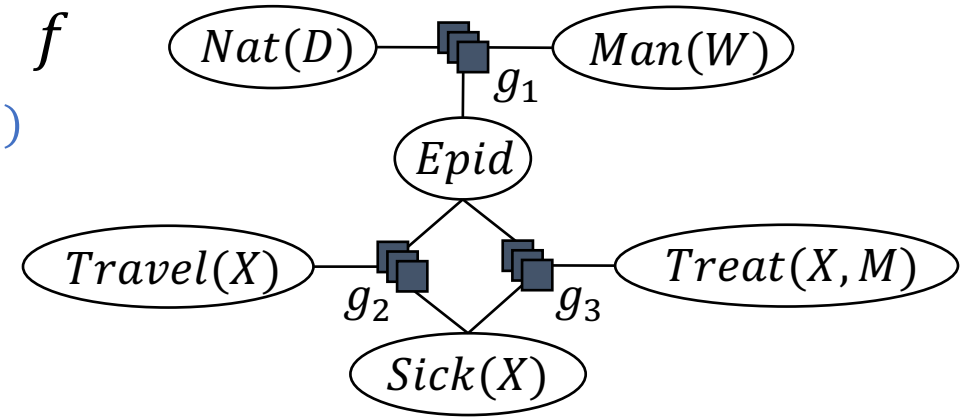
- Given model $G = \{g_i\}_{i=1}^{n}$
  - Over PRVs $rv(G)$
  - Equivalent propositional model over random variables $F = gr\big(rv(G)\big) = \{R_1, \ldots, R_N\}$ with semantics over P_F

As a way of ensuring correctness

- Semantics: Build full joint probability distribution P_G

$$P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$$

$$Z = \sum_{r_1 \in \mathcal{R}(R_1)} \cdots \sum_{r_N \in \mathcal{R}(R_N)} \prod_{f \in gr(G)} f$$



Outline: 3. Lifted Inference

A. Lifted variable elimination (LVE)
- Operators
- Algorithm
- Complexity (including first-order dtrees), completeness, tractability
- Variants

B. Lifted junction tree algorithm (LJT)
- First-order junction trees (FO jtrees)
- Algorithm
- Complexity, completeness
- Variants

C. First-order knowledge compilation (FOKC)
- Normal form, circuits
- Algorithm
- Complexity, completeness

D. Most probable assignment queries
- Distribution vs. assignment queries
- Most probable explanation (MPE) , Maximum-a-posteriori (MAP) assignments
- Changes in LVE, LJT, FOKC
- Complexity, completeness

Lifted Variable Elimination (LVE)

- Inference task: Solve an instance of the query answering problem
  - Compute an answer to a query $P(\boldsymbol{S}|\boldsymbol{T})$ given a model G representing the full joint probability distribution P_G
    - Query for a marginal (conditional) probability (distribution)
    - Avoid grounding (parts of) G

- We will concentrate on $P(S|\boldsymbol{T})$ for now
  - S single grounded PRV (no event)
- Basic idea: Eliminate all non-query terms!
  - As before
- But: Lifting of
  - Summing out
  - Multiplication
  - Absorption of $\boldsymbol{T}$

  → Lifting operators of LVE

Lifted Summing Out: Idea

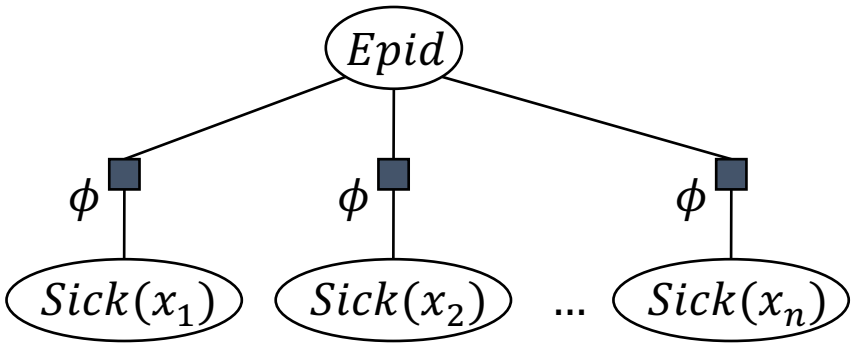
$$P(Epid)$$

$$\propto \sum_{s \in \mathcal{R}(Sick(x_1))} \phi(Epid, Sick(x_1) = s)$$

$$\cdot \sum_{s \in \mathcal{R}(Sick(x_2))} \phi(Epid, Sick(x_2) = s)$$

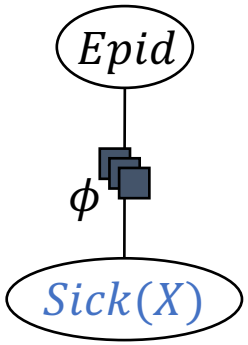
$$\cdot \ldots$$

$$\cdot \sum_{s \in \mathcal{R}(Sick(x_n))} \phi(Epid, Sick(x_n) = s)$$



$$P(Epid)$$

$$\propto \left(\sum_{s \in r(Sick(x))} \phi(Epid, Sick(x) = s) \right)^n$$



- Equivalence between ground and lifted calculations
  - Requires three preconditions for correct calculations
    - Ensuring same outcome for equivalent computations in ground case

Preconditions

- For summing out PRV A in parfactor

$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

1. PRV A under $(\mathcal{X}, C_{\mathcal{X}})$ only occurs in g, i.e.,

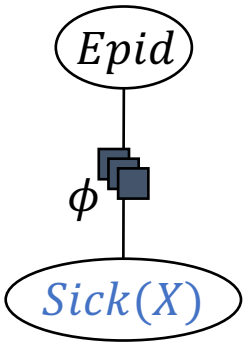
$$\forall B \in rv(G \setminus \{g\}) : gr(B_{|C}) \cap gr(A_{|(\mathcal{X},C_{\mathcal{X}})}) = \emptyset$$

   - Same as with propositional variable elimination
     - (Multiplication before addition)
   - If not the case, multiply all g containing A into one parfactor
     - How to do multiplication lifted → next operator

Preconditions

- For summing out PRV A in parfactor
$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

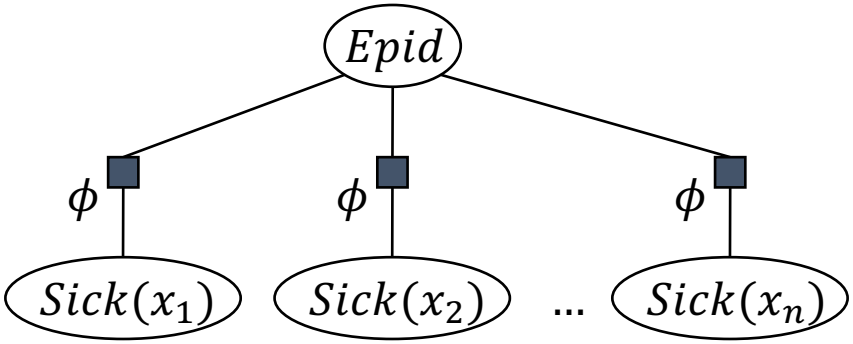
2. PRV A contains all logvars that are not singleton in $C_{\mathcal{X}}$, i.e.,
$$\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A)$$
   - If singleton: could ground logvar and remove it from g
   - E.g., $A = Epid$?
     - $(\mathcal{X}, C_{\mathcal{X}}) = \big((X), \{(x_1), \dots, (x_n)\}\big)$
     - $Epid$ does not contain X
     - Cannot eliminate $Epid$ before $Sick(X)$
       - Equivalent to ground case



Preconditions

- For summing out PRV A in parfactor
$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

2. PRV A contains all logvars that are not singleton in $C_{\mathcal{X}}$, i.e.,
$$\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A)$$
   - If singleton: could ground logvar and remove it from g
   - E.g., $A = Epid$?
     - Ground case
       - To eliminate $Epid$: Multiply all (ground) ϕ's
       - Would lead to large tree width → eliminate $Sick(x_i)$ first



Preconditions

- For summing out PRV A in parfactor

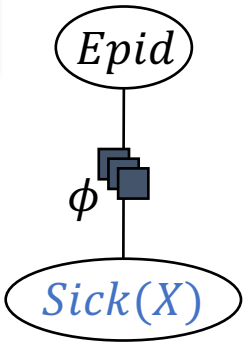
$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

2. PRV A contains all logvars that are not singleton in $C_{\mathcal{X}}$, i.e.,

$$\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A)$$

  - If singleton: could ground logvar and remove it from g
  - If not the case,
    - ~~Ground those logvars~~
    - Choose other PRV to eliminate
  - E.g., in example
    - Eliminate $A = Sick(X)$ first

Avoid grounding!
(Only a last resort)



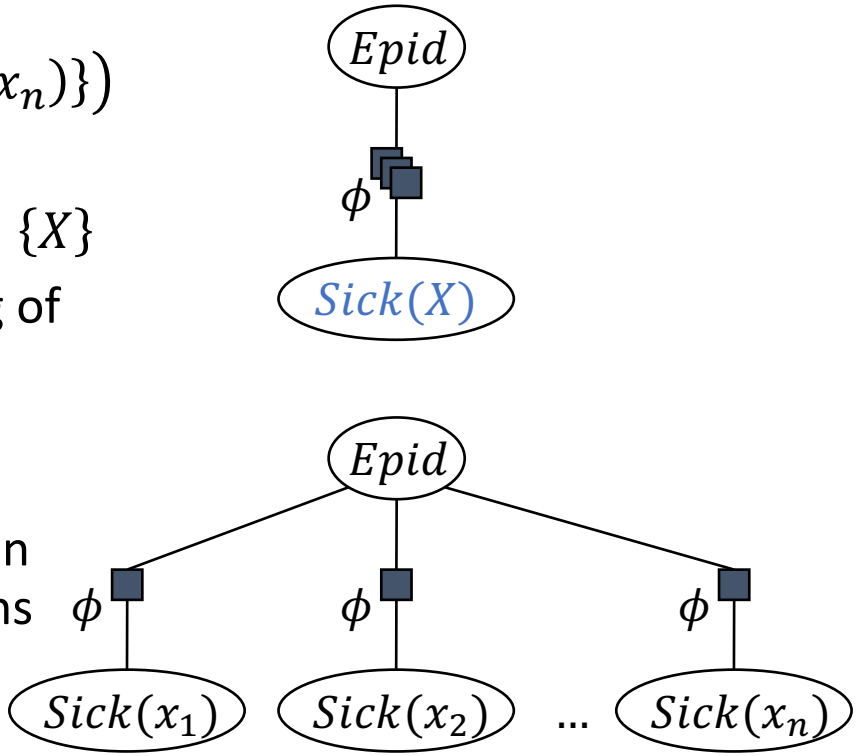
Preconditions

- For summing out PRV A in parfactor
$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

3. The logvars that occur only in A,
$$\boldsymbol{X}^{excl} = lv(A) \setminus lv(\mathcal{A} \setminus \{A\})$$
need to be count-normalised w.r.t. the remaining logvars in g,
$$\boldsymbol{X}^{com} = lv(A) \cap lv(\mathcal{A} \setminus \{A\})$$
   - Count normalisation:
     For the different possible groundings of $\boldsymbol{X}^{com}$, the same number of groundings of $\boldsymbol{X}^{excl}$ exist

Count Normalisation

- For the different possible groundings of $\boldsymbol{X}^{com}$, the same number of groundings of $\boldsymbol{X}^{excl}$ exist
- Trivial if $\boldsymbol{X}^{com} = \emptyset$:
  - E.g.,
    - $(\mathcal{X}, C_{\mathcal{X}}) = \big((X), \{(x_1), \dots, (x_n)\}\big)$
    - $\boldsymbol{X}^{com} = lv(Epid) = \emptyset$
    - $\boldsymbol{X}^{excl} = lv\big(Sick(X)\big) \setminus \emptyset = \{X\}$
    - For each possible grounding of $Epid$, which is just one, namely $Epid$, there are n groundings of X
    - One lifted sum-out operation replaces n ground operations



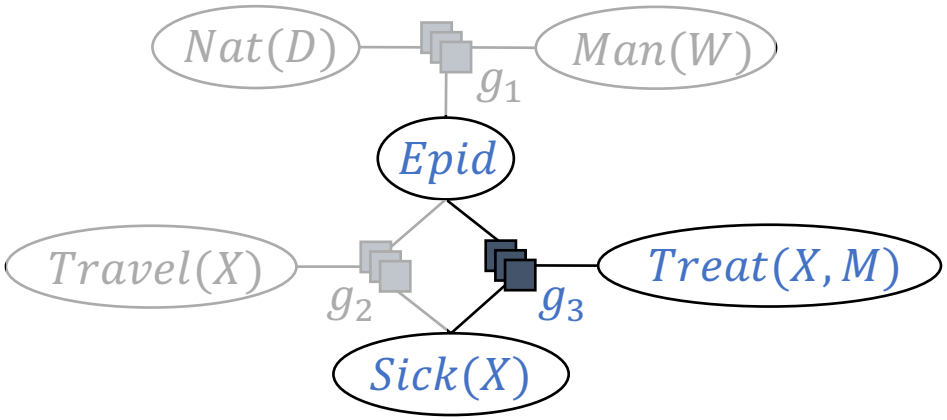
Count Normalisation

- For the different possible groundings of $\boldsymbol{X}^{com}$, the same number of groundings of $\boldsymbol{X}^{excl}$ exist
- Non-trivial case:
  - E.g., $A = Treat(X, M)$
    - With $a = alice, e = eve, b = bob : (\mathcal{X}, C_{\mathcal{X}}) = \big((X, M), \{(a, m_1), (a, m_2), (e, m_1), (e, m_2), (b, m_1), (b, m_2)\}\big)$
    - $\boldsymbol{X}^{com} = lv\big(Epid, Sick(X)\big) = \{X\}$
    - $\boldsymbol{X}^{excl} = lv\big(Treat(X, M)\big) \setminus \{X\} = \{M\}$
  - For each possible grounding of X, there must be the same number of groundings of M
    - a: 2
    - e: 2
    - b: 2

One lifted sum-out replaces for each x two ground operations

Even more strict: Require not only the same number, but the *same constants*!

$a : \{m_1, m_2\}$
$e : \{m_1, m_2\}$
$b : \{m_1, m_2\}$



Formal Definition

- More general: Given a constraint $(\mathcal{X}, C_{\mathcal{X}})$, the same number of groundings of $\boldsymbol{Y} \subseteq \boldsymbol{X}$ exist for the different possible groundings of $\boldsymbol{Z} \subseteq \boldsymbol{X} \setminus \boldsymbol{Y}$, with $\boldsymbol{X}$ the set of $\mathcal{X}$
- Count function:
  Given a constraint $(\mathcal{X}, C_{\mathcal{X}})$, for any $\boldsymbol{Y} \subseteq \mathcal{X}$ and $\boldsymbol{Z} \subseteq \mathcal{X} \setminus \boldsymbol{Y}$, the function $\text{count}_{Y|Z}(t) : C_{\mathcal{X}} \to \mathbb{N}$ is defined by

$$\text{count}_{\boldsymbol{Y}|\boldsymbol{Z}}(t) = \left|\pi_{\boldsymbol{Y}}\left(C_{\mathcal{X}} \bowtie \pi_{\boldsymbol{Z}}(\{t\})\right)\right|$$

- Count-normalisation:
  $\boldsymbol{Y}$ is count-normalised w.r.t. to $\boldsymbol{Z}$ iff $\exists n \in \mathbb{N}$ s.t.

$$\forall t \in C_{\mathcal{X}} : \text{count}_{\boldsymbol{Y}|\boldsymbol{Z}}(t) = n$$

  - Conditional count of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, denoted $\text{ncount}_{\boldsymbol{Y}|\boldsymbol{Z}}\left((\mathcal{X}, C_{\mathcal{X}})\right)$

Example

- $\text{count}_{\boldsymbol{Y}|\boldsymbol{Z}}(t) = \left|\pi_{\boldsymbol{Y}}\left(C_{\mathcal{X}} \bowtie \pi_{\boldsymbol{Z}}(\{t\})\right)\right|$
  - E.g.,
    - $\mathcal{X} = (X, M)$
    - $\boldsymbol{Y} = \{M\}$
    - $\boldsymbol{Z} = \mathcal{X} \setminus \boldsymbol{Y} = \{X\}$
    - With $a = alice, e = eve, b = bob$:
      $C_{\mathcal{X}} = \{(a, m_2), (e, m_1), (e, m_2), (b, m_1), (b, m_2)\}$

$\text{count}_{M|X}\left((a, m_2)\right)$

$\pi_X(\{(a, m_2)\}) = \{(a)\}$

$C_{X,M} \bowtie \{(a)\} = \{(a, m_2)\}$

$\pi_M(\{(a, m_2)\}) = \{(m_2)\}$

$|\{(m_2)\}| = 1$

$\text{count}_{M|X}\left((e, m_1)\right)$

$\pi_X(\{(e, m_1)\}) = \{(e)\}$

$C_{X,M} \bowtie \{(e)\} = \{(e, m_1), (e, m_2)\}$

$\pi_M(\{(e, m_1), (e, m_2)\}) = \{(m_1), (m_2)\}$

$|\{(m_1), (m_2)\}| = 2$

- Not count-normalised
  - $1 \neq 2$

Adding (a, m_1) to $C_{\mathcal{X}}$ leads to $\text{count}_{M|X}\left((a, m_2)\right) = 2$
→ M is count-normalised w.r.t. X

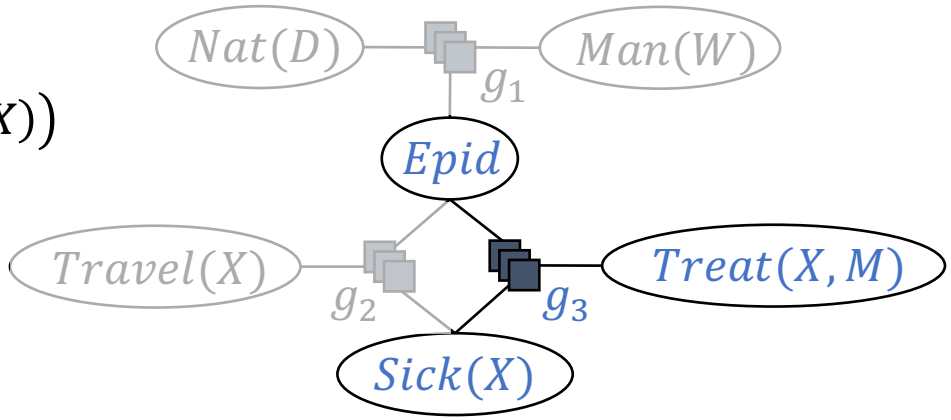
Preconditions

- For summing out PRV A in parfactor
$$g = \phi(\mathcal{A})_{|(\mathcal{X},C_{\mathcal{X}})}$$

3. The logvars that occur only in A,
$$\boldsymbol{X}^{excl} = lv(A) \setminus \left(\boldsymbol{X} \setminus lv(A)\right)$$
need to be count-normalised w.r.t. the remaining logvars in g,
$$\boldsymbol{X}^{com} = lv(A) \cap lv(\boldsymbol{X})$$
   - With $\boldsymbol{X}$ the set of $\mathcal{X}$
   - For each sequence of constants left, the same number of sequences must be eliminated
     - Only way to have the same number for exponentiation after summing-out
     - Number for exponentiation: $\mathrm{ncount}_{\boldsymbol{X}^{excl}|\boldsymbol{X}^{com}}\left((\mathcal{X}, C_{\mathcal{X}})\right)$

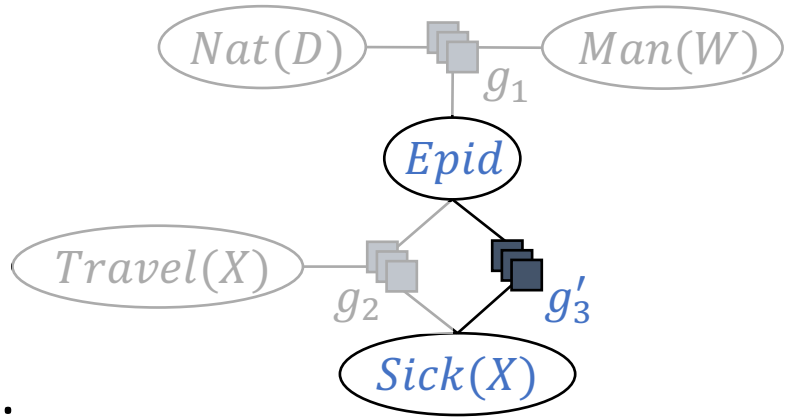
Example

- Summing out $Treat(X, M)$ in
  $g_3 = \phi_3\big(Epid, Sick(X), Treat(X, M)\big)_{((X,M),\mathcal{D}(X)\times\mathcal{D}(M))}$
- Preconditions:
  - $Treat(X, M)$ occurs only in g_3
  - $Treat(X, M)$ contains all logvars in g_3
  - M is count-normalised w.r.t. X,
    $$\text{ncount}_{M|X}\big((\mathcal{X}, C_{\mathcal{X}})\big) = 2$$
- Output:
  - $\phi_3'\big(Epid, Sick(X)\big)_{|(X,\mathcal{D}(X))}$



Example

- Output:
  - $\phi_3'(Epid, Sick(X))_{|(X,\mathcal{D}(X))}$
  - For each assignment to $Epid, Sick(X)$, sum over range values of $Treat(X, M)$…

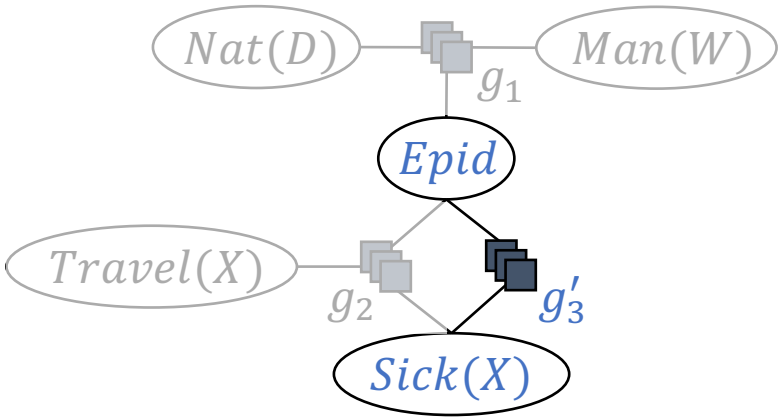


$Epid$	$Sick(X)$	ϕ_3^*	
$false$	$false$	5	+
$false$	$true$	10	+
$true$	$false$	10	+
$true$	$true$	11	+

$Epid$	$Sick(X)$	$Treat(X,M)$	ϕ_3
$false$	$false$	$false$	5
$false$	$false$	$true$	0
$false$	$true$	$false$	4
$false$	$true$	$true$	6
$true$	$false$	$false$	4
$true$	$false$	$true$	6
$true$	$true$	$false$	2
$true$	$true$	$true$	9

Example

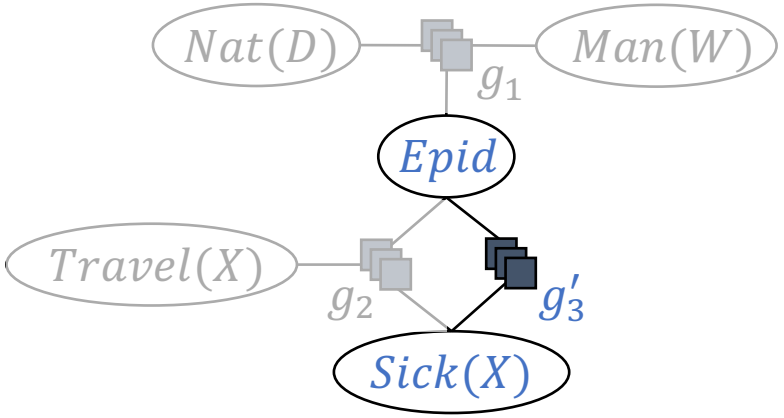
- Output:
  - $\phi_3'\big(Epid, Sick(X)\big)_{|(X,\mathcal{D}(X))}$
  - … and raise to the power of $\mathrm{ncount}_{M|X}\big((\mathcal{X}, C_{\mathcal{X}})\big) = 2$

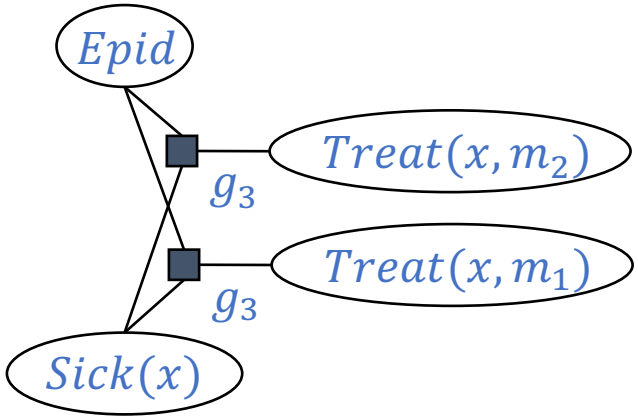


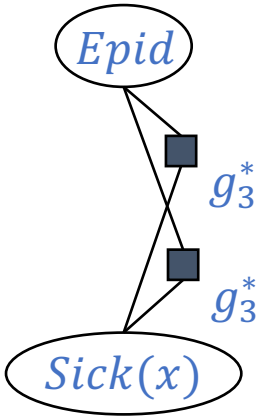
$Epid$	$Sick(X)$	ϕ_3'
$false$	$false$	5^2
$false$	$true$	10^2
$true$	$false$	10^2
$true$	$true$	11^2

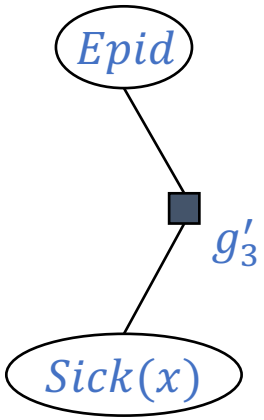
Example

- Output:
  - $\phi_3'(Epid, Sick(X))_{|(X,\mathcal{D}(X))}$
  - Equivalent to ground case
    - For each x, the following sum-out operations occur:









Initial model

After sum-out

After multiply

All factors contain the same arguments after sum-out

Lifted Summing Out: Operator

- Inputs:
  - Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
  - PRV A_i occurring in $\mathcal{A}$ for summing out
- Preconditions:
  - $\forall B \in rv(G \setminus \{g\}) : gr(B_{|C}) \cap gr(A_{|(\mathcal{X}, C_{\mathcal{X}})}) = \emptyset$
  - $\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A)$
  - $\boldsymbol{X}^{excl} = lv(A) \setminus (\boldsymbol{X} \setminus lv(A))$ count-normalised w.r.t. $\boldsymbol{X}^{com} = lv(A) \cap \boldsymbol{X}$ in C, with $\boldsymbol{X}$ the set of $\mathcal{X}$, i.e., $r = \text{ncount}_{\boldsymbol{X}^{excl}|\boldsymbol{X}^{com}}(C)$ exists
- Output: $\phi'(\mathcal{A}')_{|C'}$ with $C' = (\mathcal{X}', C_{\mathcal{X}'})$
  - $\mathcal{A}' = (A_1, \dots, A_{i-1}) \circ (A_{i+1}, \dots, A_n)$
  - $\mathcal{X}' = \pi_{\boldsymbol{X}^{com}}(\mathcal{X})$
  - $C_{\mathcal{X}'} = \pi_{\boldsymbol{X}^{com}}(C_{\mathcal{X}})$
  - For each assignment $\boldsymbol{a}' = (\dots, a_{i-1}, a_{i+1}, \dots)$ to $\mathcal{A}'$,

Concatenation of two sequences

$$\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \left(\sum_{a_i \in \mathcal{R}(A_i)} Mul(A_i, a_i) \phi(\dots, a_{i-1}, a_i, a_{i+1}, \dots) \right)^r$$

- Postcondition:

$$P_{G \setminus \{g\} \cup \{\text{sum-out}(g, A_i)\}} = \sum_{gr(A_{i|C})} P_G$$

Why $Mul(A_i, a_i)$?

Sum-out with CRVs

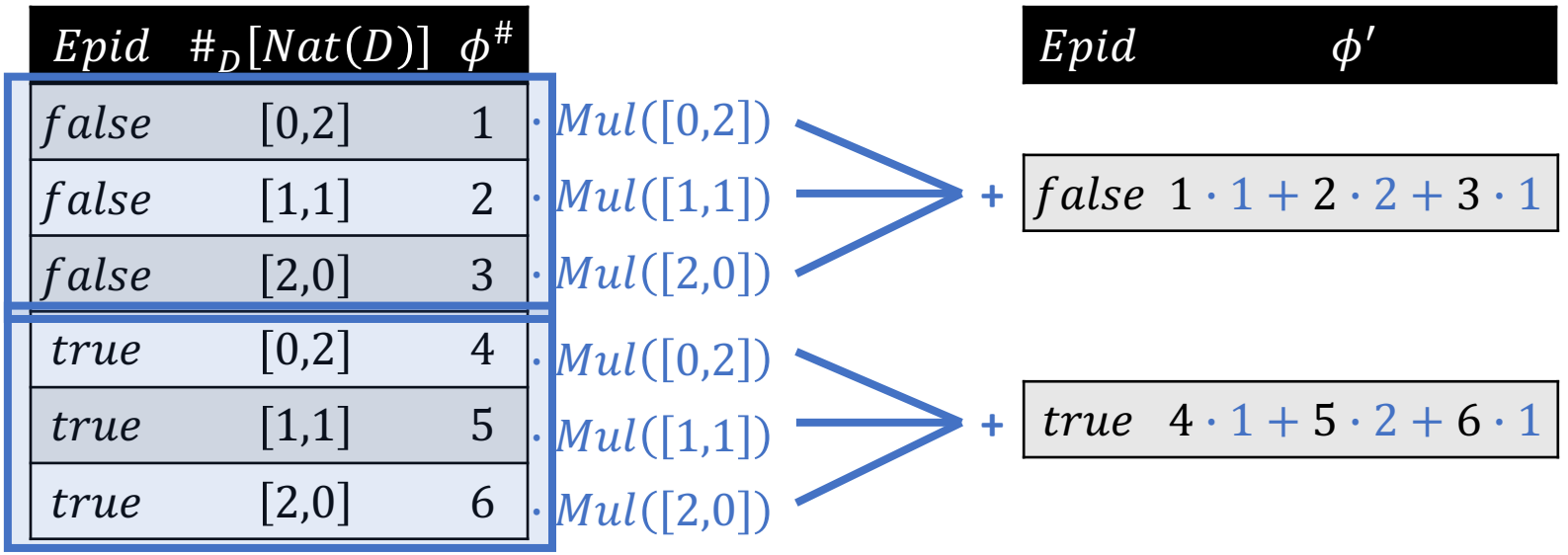
- Range values of CRVs, i.e., histograms can encode multiple assignments at once
- E.g., CRV: $\#_E[Epid(E)]$
  - Range values
    $[0,4], [1,3], [2,2], [3,1], [4,0]$
    $1 \quad 4 \quad 6 \quad 4 \quad 1$
    # of assignments encoded

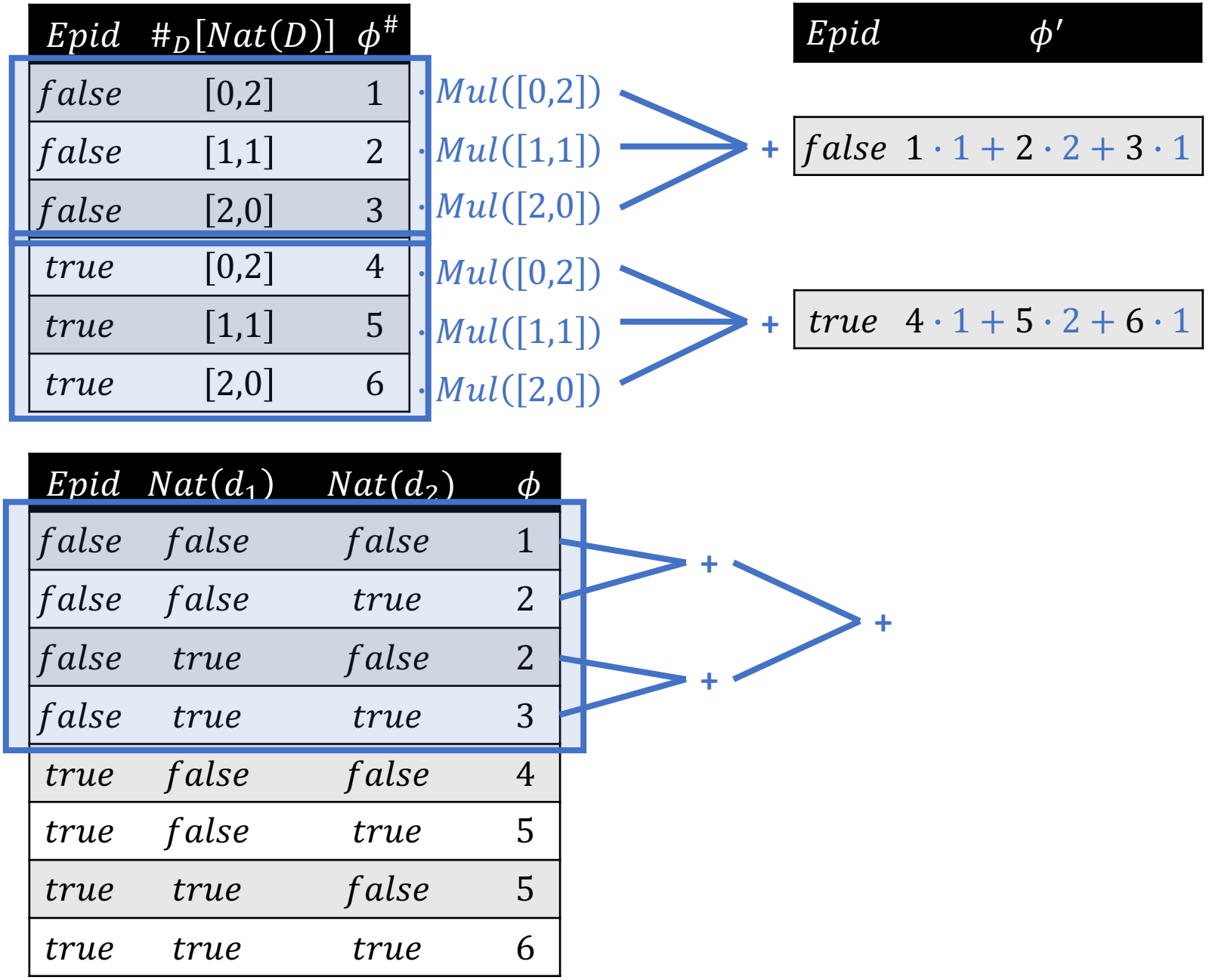
$\#_E[Epid(E)]$	g'
[0,4]	8
[1,3]	6
[2,2]	4
[3,1]	2
[4,0]	0

$Epid_1$	$Epid_2$	$Epid_3$	$Epid_4$	g
$false$	$false$	$false$	$false$	8
$false$	$false$	$false$	$true$	6
$false$	$false$	$true$	$false$	6
$false$	$false$	$true$	$true$	4
$false$	$true$	$false$	$false$	6
$false$	$true$	$false$	$true$	4
$false$	$true$	$true$	$false$	4
$false$	$true$	$true$	$true$	2
$true$	$false$	$false$	$false$	6
$true$	$false$	$false$	$true$	4
$true$	$false$	$true$	$false$	4
$true$	$false$	$true$	$true$	2
$true$	$true$	$false$	$false$	4
$true$	$true$	$false$	$true$	2
$true$	$true$	$true$	$false$	2
$true$	$true$	$true$	$true$	0

Sum-out with CRVs

- Summing over the range of CRVs sums over histograms, i.e., over multiple assignments
  - Each histogram h represents $Mul(h)$ assignments
  - E.g., sum out $\#_D[Nat(D)]$ in $\phi^{\#}(Epid, \#_D[Nat(D)])$
    - All $gr(Nat(D))$ summed out at once




Epid
$\#_D[Nat(D)]$
$\phi^{\#}$
false [0,2] 1 · Mul([0,2])
false [1,1] 2 · Mul([1,1])
false [2,0] 3 · Mul([2,0])
true [0,2] 4 · Mul([0,2])
true [1,1] 5 · Mul([1,1])
true [2,0] 6 · Mul([2,0])
+
Epid
ϕ'
false 1 · 1 + 2 · 2 + 3 · 1
true 4 · 1 + 5 · 2 + 6 · 1
Epid
$Nat(d_1)$
$Nat(d_2)$
ϕ
false false false 1
false false true 2
false true false 2
false true true 3
true false false 4
true false true 5
true true false 5
true true true 6
+
+
+

$Epid$	$\#_D[Nat(D)]$	$\phi^{\#}$	
$false$	[0,2]	1	$\cdot Mul([0,2])$
$false$	[1,1]	2	$\cdot Mul([1,1])$
$false$	[2,0]	3	$\cdot Mul([2,0])$
$true$	[0,2]	4	$\cdot Mul([0,2])$
$true$	[1,1]	5	$\cdot Mul([1,1])$
$true$	[2,0]	6	$\cdot Mul([2,0])$

$Epid$	ϕ'
$false$	$1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1$
$true$	$4 \cdot 1 + 5 \cdot 2 + 6 \cdot 1$

$Epid$	$Nat(d_1)$	$Nat(d_2)$	ϕ
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	2
$false$	$true$	$true$	3
$true$	$false$	$false$	4
$true$	$false$	$true$	5
$true$	$true$	$false$	5
$true$	$true$	$true$	6

$Epid$	ϕ'
$false$	$1 + 2 + 2 + 3$
$true$	$4 + 5 + 5 + 6$

Lifted Summing Out: Operator

- Inputs:
  - Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
  - PRV A_i occurring in $\mathcal{A}$ for summing out
- Preconditions:
  - $\forall B \in rv(G \setminus \{g\}) : gr(B_{|C}) \cap gr(A_{|(\mathcal{X}, C_{\mathcal{X}})}) = \emptyset$
  - $\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A)$
  - $\boldsymbol{X}^{excl} = lv(A) \setminus (\boldsymbol{X} \setminus lv(A))$ count-normalised w.r.t. $\boldsymbol{X}^{com} = lv(A) \cap \boldsymbol{X}$ in C, with $\boldsymbol{X}$ the set of $\mathcal{X}$, i.e., $r = \text{ncount}_{\boldsymbol{X}^{excl}|\boldsymbol{X}^{com}}(C)$ exists
- Output: $\phi'(\mathcal{A}')_{|C'}$ with $C' = (\mathcal{X}', C_{\mathcal{X}'})$
  - $\mathcal{A}' = (A_1, \dots, A_{i-1}) \circ (A_{i+1}, \dots, A_n)$
  - $\mathcal{X}' = \pi_{\boldsymbol{X}^{com}}(\mathcal{X})$
  - $C_{\mathcal{X}'} = \pi_{\boldsymbol{X}^{com}}(C_{\mathcal{X}})$
  - For each assignment $\boldsymbol{a}' = (\dots, a_{i-1}, a_{i+1}, \dots)$ to $\mathcal{A}'$,

$$\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \left(\sum_{a_i \in \mathcal{R}(A_i)} Mul(A_i, a_i) \, \phi(\dots, a_{i-1}, a_i, a_{i+1}, \dots) \right)^r$$

$Mul(A_i, a_i)$ makes sum-out of CRVs correct. If A_i no CRV, $Mul(A_i, a_i) = 1$

- Postcondition:

$$P_{G \setminus \{g\} \cup \{\text{sum-out}(g, A_i)\}} = \sum_{gr(A_{i|C})} P_G$$

Lifted Summing Out

- Summing out transforms the current model G
  - Removes a PRV from $rv(G)$
- Preconditions already act as a filter on possible $\mathrm{sum}{-}\mathrm{out}$ operations
- Effect:
  - Number of logvars per PRV that is eliminated is decreasing over the whole LVE run for one query
  - Until only propositional random variables are left

    → Standard variable elimination

Lifted Multiplication

- Operator for multiplication as an ”enabler” for $\mathrm{sum}-\mathrm{out}$ operator
  - Precondition 1: PRV to sum out may only appear in one parfactor
- Multiply two parfactors
  - Still a join of over arguments and a product of potentials
  - But, CAUTION:
    Since two parfactors represent two (different) sets of factors, lifted multiplication has to work as a representative multiplication for those two sets

Lifted Multiplication: Trivial Case

- 1-to-1 correspondence between the ground factors of each parfactor
  - E.g., $\phi_1(S(X)) \cdot \phi_2(S(X), A(X))$

Each grounding of X in $gr\left(\phi_1(S(X))\right)$ interacts with one corresponding grounding of X in $gr\left(\phi_2(S(X), A(X))\right)$

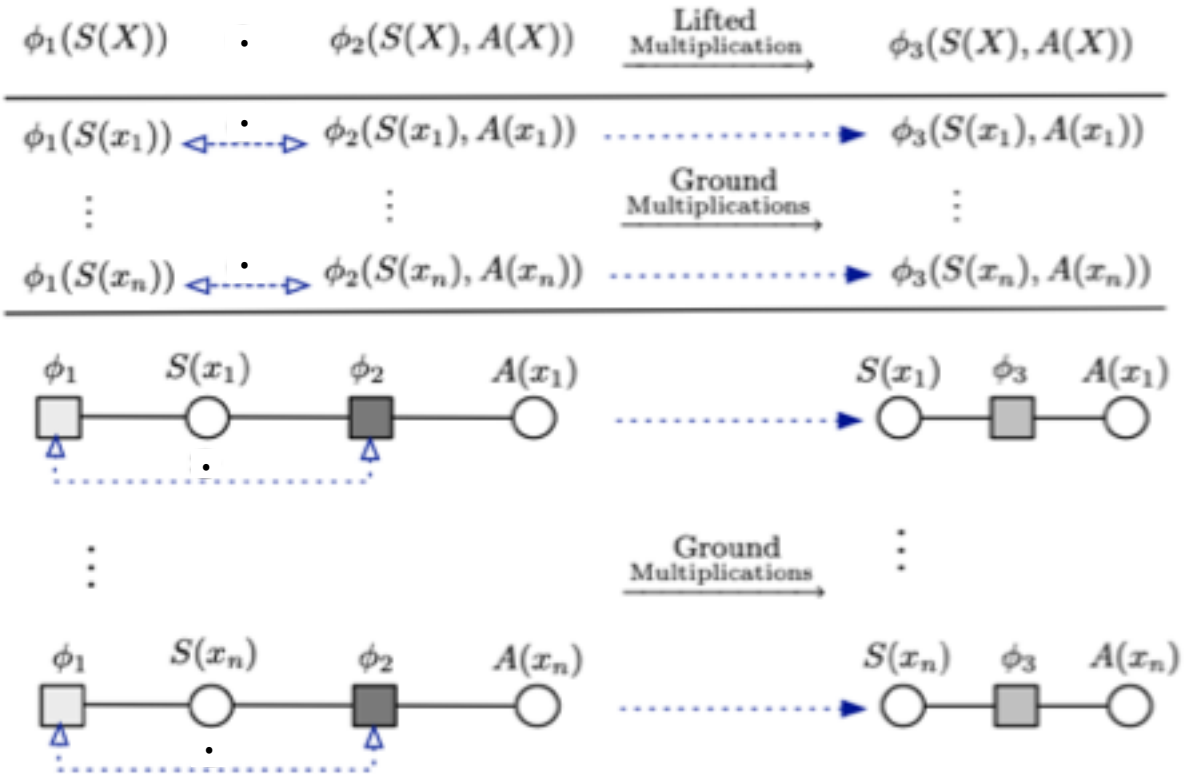


Figure taken from: Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013.

Lifted Multiplication: More General

- 1-to-m correspondence between the ground factors of each parfactor
  - Scaling necessary
  - E.g., $\phi_1\big(S(X)\big) \cdot \phi_2\big(S(X), F(X,Y)\big)$

Each grounding of X in $gr\Big(\phi_1\big(S(X)\big)\Big)$ interacts with m corresponding groundings of X, Y in $gr\Big(\phi_2\big(S(X), F(X,Y)\big)\Big)$

Distribute $\phi_1\big(S(X)\big)$ into m factors proportionally

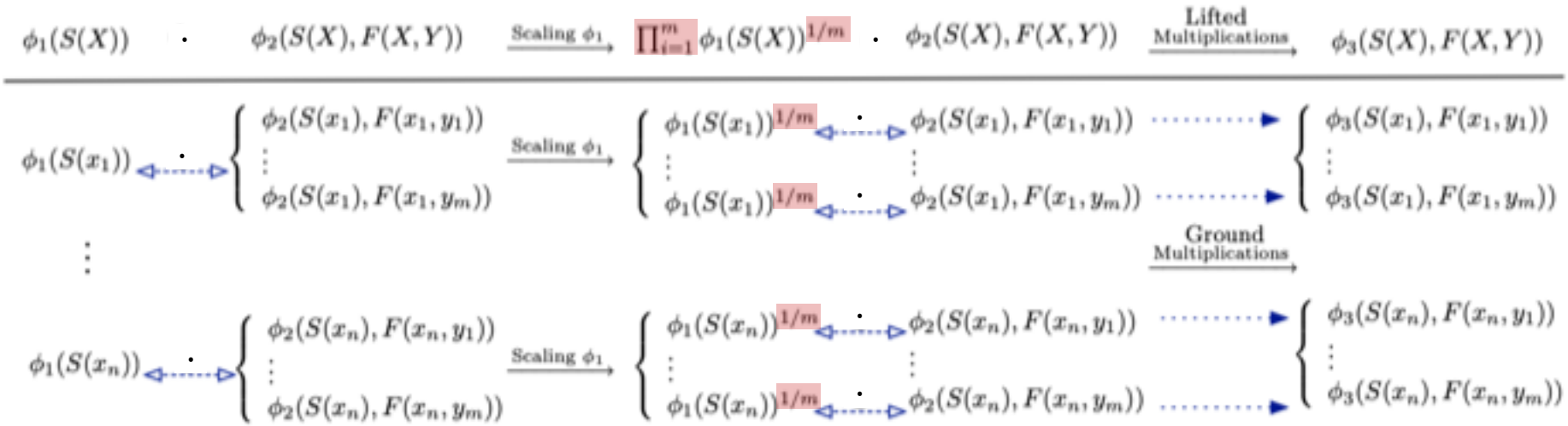


Figure taken from: Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013.

Lifted Multiplication: General Case

- n-to-m correspondence between the ground factors of each parfactor
  - Scaling necessary in both directions
  - E.g., $\phi_1(S(X), T(X,Z)) \cdot \phi_2\big(S(X), F(X,Y)\big)$
    - $\mathcal{D}(X) = \{x_1, \dots, x_k\}, \mathcal{D}(Z) = \{z_1, \dots, z_n\}, \mathcal{D}(Y) = \{y_1, \dots, y_m\}$
    - Each grounding of X, Z in ϕ_1 interacts with m groundings of X, Y in ϕ_2
    - Each grounding of X, Y in ϕ_2 interacts with n groundings of X, Z in ϕ_1
    - Scaling:

$$\prod_{i=1}^{m} \Big(\phi_1\big(S(X), T(X,Z)\big)\Big)^{\frac{1}{m}} \cdot \prod_{i=1}^{n} \Big(\phi_2\big(S(X), F(X,Y)\big)\Big)^{\frac{1}{n}}$$

Distribute $\phi_1\big(S(X), T(X,Z)\big)$ into m factors and $\phi_1\big(S(X), F(X,Y)\big)$ into n factors proportionally

Lifted Multiplication: Operator

- Inputs:
  - Parfactor $g_1 = \phi_1(\mathcal{A}_1)_{|C_1}, C_1 = (\mathcal{X}_1, C_{\mathcal{X}_1})$
  - Parfactor $g_2 = \phi_2(\mathcal{A}_2)_{|C_2}, C_2 = (\mathcal{X}_2, C_{\mathcal{X}_2})$
  - One-to-one substitution $\theta = \{\boldsymbol{Z}_1 \to \boldsymbol{Z}_2\}$ between the logvars of the shared PRVs in g_1 and g_2
- Preconditions:
  - For $i = 1,2$: $\boldsymbol{Y}_i = \boldsymbol{X}_i \setminus \boldsymbol{Z}_i$ count-normalised w.r.t. $\boldsymbol{Z}_i$ in C_i, with $\boldsymbol{X}_i$ the set of $\mathcal{X}_i$, i.e., $r_i = \mathrm{ncount}_{\boldsymbol{Y}_i|\boldsymbol{Z}_i}(C_i)$ exists
- Output: $\phi(\mathcal{A})_{|C}$ with $C = (\mathcal{X}, C_{\mathcal{X}})$
  - $\mathcal{A} = \mathcal{A}_1\theta \bowtie \mathcal{A}_2$
  - $\mathcal{X} = \mathcal{X}_1\theta \bowtie \mathcal{X}_2$
  - $C_{\mathcal{X}} = C_{\mathcal{X}_1\theta} \bowtie C_{\mathcal{X}_2}$
  - For each assignment $\boldsymbol{a}$ to $\mathcal{A}$, with $\boldsymbol{a}_1 = \pi_{\mathcal{A}_1\theta}(\boldsymbol{a})$ and $\boldsymbol{a}_2 = \pi_{\mathcal{A}_2\theta}(\boldsymbol{a})$

$$\phi(\boldsymbol{a}) = \left(\phi_1(\boldsymbol{a}_1)\right)^{\frac{1}{r_2}} \cdot \left(\phi_2(\boldsymbol{a}_2)\right)^{\frac{1}{r_1}}$$

- Postcondition:

$$G \sim G \setminus \{g_1, g_2\} \cup \{\mathrm{multiply}(g_1, g_2, \theta)\}$$

Operator does not assume that logvars with the same applicable constants share the same name

Lifted Multiplication: Example

$$g_1 \cdot g_2$$
$$= \phi_1\big(Epid, Travel(X)\big) \cdot \phi_2\big(Travel(X), Sick(X)\big)$$
$$= \phi\big(Epid, Travel(X), Sick(X)\big)$$

- 1-to-1
  - $\boldsymbol{X}_1 = lv(g_1) = \{X\} = lv(g_2) = \boldsymbol{X}_2$
  - $\boldsymbol{Z}_1 = lv\big(Travel(X)\big) = \{X\} = lv\big(Travel(X)\big) = \boldsymbol{Z}_2$
    - No substitution necessary to align logvar names
  - $\boldsymbol{Y}_1 = \boldsymbol{X}_1 \setminus \boldsymbol{Z}_1 = \emptyset$ count-normalised w.r.t. $\boldsymbol{Z}_1 = \{X\}$
  - $\boldsymbol{Y}_2 = \boldsymbol{X}_2 \setminus \boldsymbol{Z}_2 = \emptyset$ count-normalised w.r.t. $\boldsymbol{Z}_2 = \{X\}$
    - No scaling necessary

Lifted Multiplication: Example

$$g_1 \cdot g_2$$
$$= \phi_1\big(Epid, Travel(X)\big) \cdot \phi_2\big(Travel(X), Sick(X)\big)$$
$$= \phi\big(Epid, Travel(X), Sick(X)\big)$$

$Epid$	$Travel(X)$	$Sick(X)$	ϕ
$false$	$false$	$false$	$1 \cdot 5$
$false$	$false$	$true$	
$false$	$true$	$false$	
$false$	$true$	$true$	
$true$	$false$	$false$	
$true$	$false$	$true$	
$true$	$true$	$false$	
$true$	$true$	$true$	

$\cdot$

$\pi_{Epid,Travel(X)}$

$Epid$	$Travel(X)$	ϕ_1
$false$	$false$	1
$false$	$true$	2
$true$	$false$	3
$true$	$true$	4

$\pi_{Travel(X),Sick(X)}$

$Travel(X)$	$Sick(X)$	ϕ_2
$false$	$false$	5
$false$	$true$	6
$true$	$false$	7
$true$	$true$	8

Lifted Multiplication: Example

$$g_1 \cdot g_2$$
$$= \phi_1\big(Epid, Travel(X)\big) \cdot \phi_2\big(Travel(X), Sick(X)\big)$$
$$= \phi\big(Epid, Travel(X), Sick(X)\big)$$

$Epid$	$Travel(X)$	$Sick(X)$	ϕ
$false$	$false$	$false$	$1 \cdot 5$
$false$	$false$	$true$	$1 \cdot 6$
$false$	$true$	$false$	
$false$	$true$	$true$	
$true$	$false$	$false$	
$true$	$false$	$true$	
$true$	$true$	$false$	
$true$	$true$	$true$	

$\cdot$

$\pi_{Epid,Travel(X)}$

$Epid$	$Travel(X)$	ϕ_1
$false$	$false$	1
$false$	$true$	2
$true$	$false$	3
$true$	$true$	4

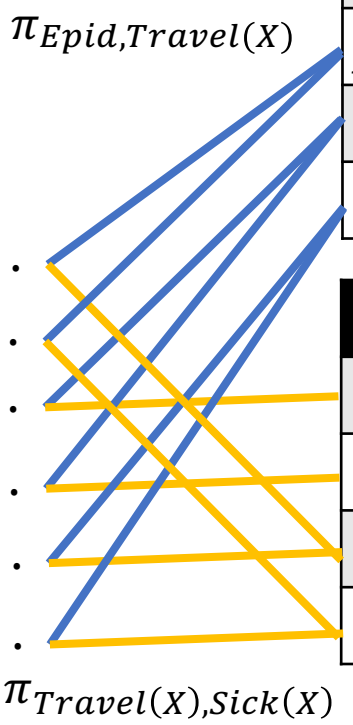
$\pi_{Travel(X),Sick(X)}$

$Travel(X)$	$Sick(X)$	ϕ_2
$false$	$false$	5
$false$	$true$	6
$true$	$false$	7
$true$	$true$	8

Lifted Multiplication: Example

$$g_1 \cdot g_2$$
$$= \phi_1\big(Epid, Travel(X)\big) \cdot \phi_2\big(Travel(X), Sick(X)\big)$$
$$= \phi\big(Epid, Travel(X), Sick(X)\big)$$

$Epid$	$Travel(X)$	$Sick(X)$	ϕ
$false$	$false$	$false$	$1 \cdot 5$
$false$	$false$	$true$	$1 \cdot 6$
$false$	$true$	$false$	$2 \cdot 7$
$false$	$true$	$true$	$2 \cdot 8$
$true$	$false$	$false$	$3 \cdot 5$
$true$	$false$	$true$	$3 \cdot 6$
$true$	$true$	$false$	$4 \cdot 7$
$true$	$true$	$true$	$4 \cdot 8$



$Epid$	$Travel(X)$	ϕ_1
$false$	$false$	1
$false$	$true$	2
$true$	$false$	3
$true$	$true$	4

$Travel(X)$	$Sick(X)$	ϕ_2
$false$	$false$	5
$false$	$true$	6
$true$	$false$	7
$true$	$true$	8

Lifted Multiplication: Example

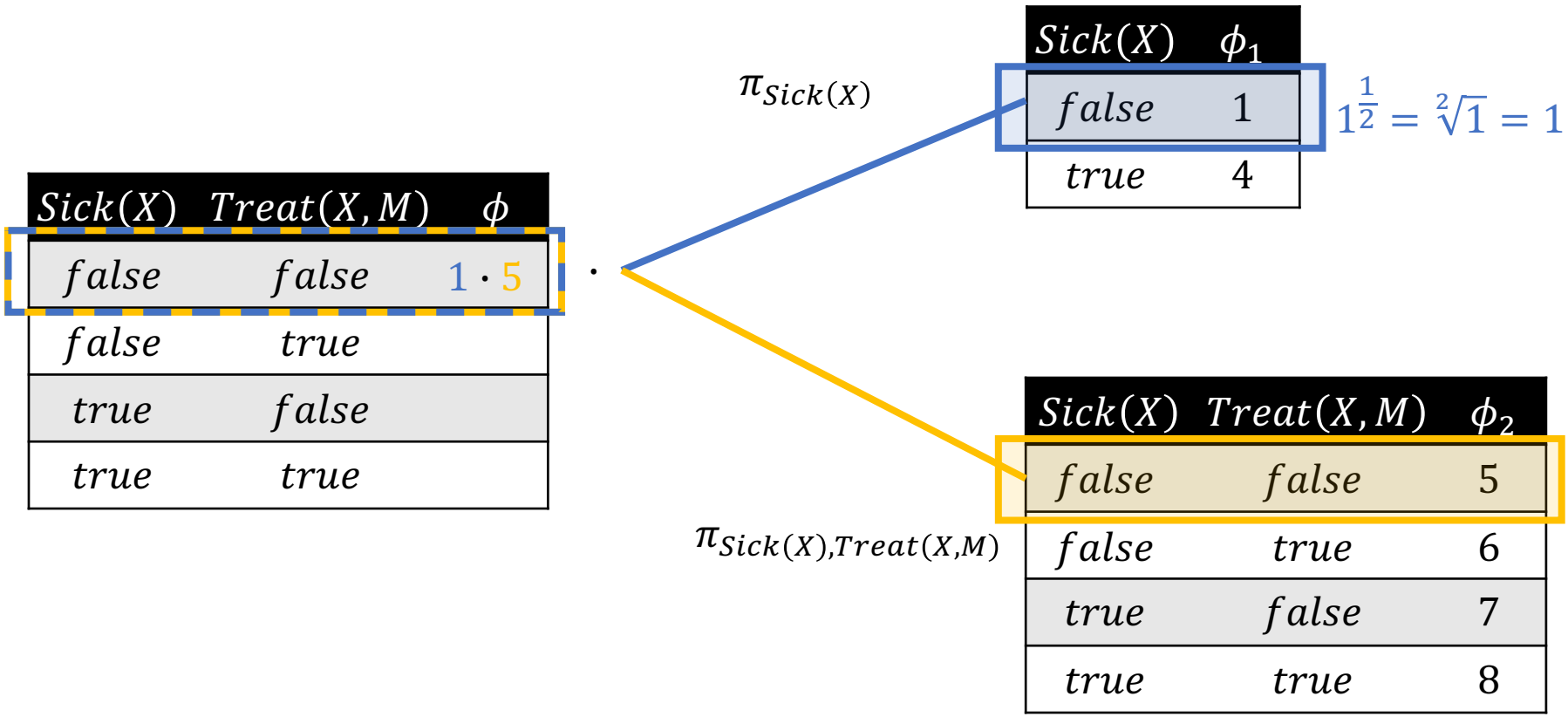
$$g_1 \cdot g_2 = \phi_1\big(Sick(X)\big) \cdot \phi_2\big(Sick(X), Treat(X,M)\big)$$
$$= \phi\big(Sick(X), Treat(X,M)\big)$$

- $\top$ constraints with $|\mathcal{D}(M)| = 2$

- 1-to-m
  - $\boldsymbol{X}_1 = lv(g_1) = \{X\}$
  - $\boldsymbol{X}_2 = lv(g_2) = \{X, M\}$
  - $\boldsymbol{Z}_1 = lv\big(Sick(X)\big) = \{X\} = lv\big(Sick(X)\big) = \boldsymbol{Z}_2$
    - No substitution necessary to align logvar names
  - $\boldsymbol{Y}_1 = \boldsymbol{X}_1 \setminus \boldsymbol{Z}_1 = \emptyset$ count-normalised w.r.t. $\boldsymbol{Z}_1 = \{X\}$
  - $\boldsymbol{Y}_2 = \boldsymbol{X}_2 \setminus \boldsymbol{Z}_2 = \{M\}$ count-normalised w.r.t. $\boldsymbol{Z}_2 = \{X\}$
    - Scaling necessary: $r_2 = \mathrm{ncount}_{\boldsymbol{Y}_2|\boldsymbol{Z}_2}(C_2) = 2$
      - Scaling of potentials in g_1

Lifted Multiplication: Example

$$g_1 \cdot g_2 = \phi_1\big(Sick(X)\big) \cdot \phi_2\big(Sick(X), Treat(X,M)\big)$$
$$= \phi\big(Sick(X), Treat(X,M)\big)$$



Lifted Multiplication: Example

$$g_1 \cdot g_2 = \phi_1\big(Sick(X)\big) \cdot \phi_2\big(Sick(X), Treat(X,M)\big)$$
$$= \phi\big(Sick(X), Treat(X,M)\big)$$

$Sick(X)$	$Treat(X,M)$	ϕ
$false$	$false$	$1 \cdot 5$
$false$	$true$	$1 \cdot 6$
$true$	$false$	$2 \cdot 7$
$true$	$true$	$2 \cdot 8$

$\cdot$
$\cdot$
$\cdot$

$\pi_{Sick(X)}$

$Sick(X)$	ϕ_1
$false$	1
$true$	4

$1^{\frac{1}{2}} = \sqrt[2]{1} = 1$

$4^{\frac{1}{2}} = \sqrt[2]{4} = 2$

$Sick(X)$	$Treat(X,M)$	ϕ_2
$false$	$false$	5
$false$	$true$	6
$true$	$false$	7
$true$	$true$	8

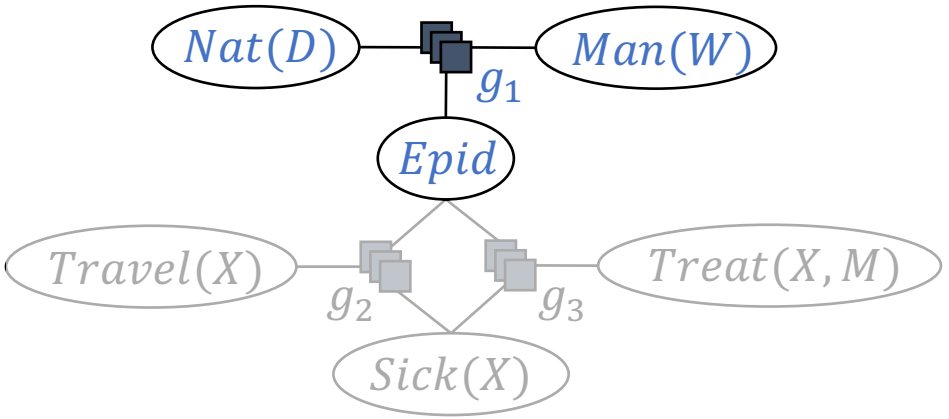
$\pi_{Sick(X),Treat(X,M)}$

Count Conversion

- Recall Precondition 2 of sum−out operator
  - PRV to sum out has to contain all logvars of a parfactor

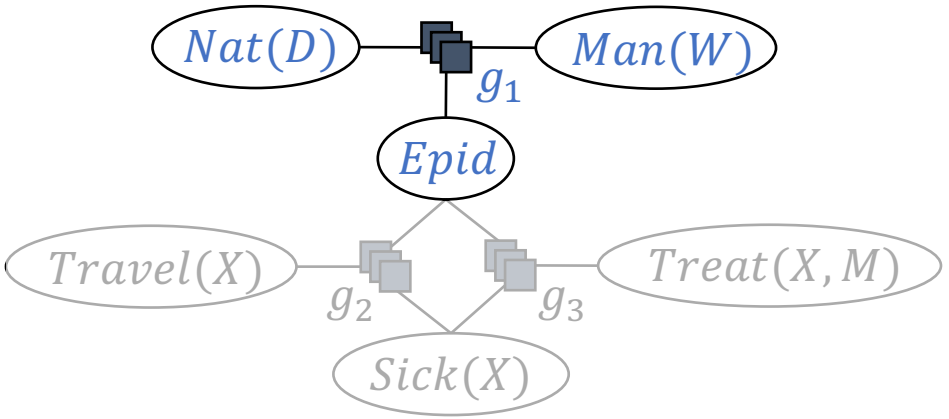
- What if no PRV contains all logvars?
  - E.g.,
    - $g_1 = \phi_1(Epid, Man(W), Nat(D))$

  - Ground a logvar?



Grounding?

- Grounding a logvar
  - E.g., D, with $\mathcal{D}(D) = \{d_1, d_2\}$
    - $g_1 = \phi_1(Epid, Man(W), Nat(D))$
  - Grounding:
    - $g_{11} = \phi_1(Epid, Man(W), Nat(d_1))$
    - $g_{12} = \phi_1(Epid, Man(W), Nat(d_2))$
  - Now, $Man(W)$ contains all logvars
  - But, to sum-out
    - Multiply g_{11}, g_{12} to fulfil Precondition 1



Grounding?

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

Grounding?

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ
$false$	$false$	$false$	$false$	$1 \cdot 1$
$false$	$false$	$false$	$true$	$1 \cdot 2$
$false$	$false$	$true$	$false$	$2 \cdot 1$
$false$	$false$	$true$	$true$	$2 \cdot 2$
⋮	⋮	⋮	⋮	⋮

Symmetries Within

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ
$false$	$false$	$false$	$false$	$1 \cdot 1$
$false$	$false$	$false$	$true$	$1 \cdot 2$
$false$	$false$	$true$	$false$	$2 \cdot 1$
$false$	$false$	$true$	$true$	$2 \cdot 2$
⋮	⋮	⋮	⋮	⋮

$\#_{true}$ $\#_{false}$ $[0,2]$ $2^0 \cdot 1^2$

Symmetries Within

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
false	*false*	*false*	1
false	*false*	*true*	2
⋮	⋮	⋮	⋮

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
false	*false*	*false*	1
false	*false*	*true*	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ	$\#_{true}, \#_{false}$	
false	*false*	*false*	*false*	$1 \cdot 1$	[0,2]	$2^0 \cdot 1^2$
false	*false*	*false*	*true*	$1 \cdot 2$	[1,1]	$2^1 \cdot 1^1$
false	*false*	*true*	*false*	$2 \cdot 1$		
false	*false*	*true*	*true*	$2 \cdot 2$		
⋮	⋮	⋮	⋮	⋮		

Symmetries Within

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ	$\#_{true}$	$\#_{false}$
$false$	$false$	$false$	$false$	$1 \cdot 1$	[0,2]	$2^0 \cdot 1^2$
$false$	$false$	$false$	$true$	$1 \cdot 2$	[1,1]	$2^1 \cdot 1^1$
$false$	$false$	$true$	$false$	$2 \cdot 1$	[1,1]	$2^1 \cdot 1^1$
$false$	$false$	$true$	$true$	$2 \cdot 2$		
⋮	⋮	⋮	⋮	⋮		

Symmetries Within

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$

$Epid$	$Man(W)$	$Nat(d_1)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

•

$Epid$	$Man(W)$	$Nat(d_2)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ	$\#_{true}$ $\#_{false}$	
$false$	$false$	$false$	$false$	$1 \cdot 1$	[0,2]	$2^0 \cdot 1^2$
$false$	$false$	$false$	$true$	$1 \cdot 2$	[1,1]	$2^1 \cdot 1^1$
$false$	$false$	$true$	$false$	$2 \cdot 1$	[1,1]	$2^1 \cdot 1^1$
$false$	$false$	$true$	$true$	$2 \cdot 2$	[2,0]	$2^2 \cdot 1^0$
⋮	⋮	⋮	⋮	⋮		

Encoding with a CRV

- $g_{11} \cdot g_{12}$
  $= \phi_1\big(Epid, Man(W), Nat(d_1)\big) \cdot \phi_1\big(Epid, Man(W), Nat(d_2)\big)$
  $= \phi\big(Epid, Man(W), Nat(d_1), Nat(d_2)\big)$
  $= \phi_1'(Epid, Man(W), \#_D[Nat(D)])$

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ
$false$	$false$	$false$	$false$	$1 \cdot 1$
$false$	$false$	$false$	$true$	$1 \cdot 2$
$false$	$false$	$true$	$false$	$2 \cdot 1$
$false$	$false$	$true$	$true$	$2 \cdot 2$
⋮	⋮	⋮	⋮	⋮

$\#_{true}$ $\#_{false}$

$[0,2]$	$2^0 \cdot 1^2$
$[1,1]$	$2^1 \cdot 1^1$
$[1,1]$	$2^1 \cdot 1^1$
$[2,0]$	$2^2 \cdot 1^0$

$Epid$	$Man(W)$	$Nat(D)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
⋮	⋮	⋮	⋮

$Epid$	$Man(W)$	$\#_D[Nat(D)]$	ϕ_1'
$false$	$false$	$[0,2]$	$2^0 \cdot 1^2$
$false$	$false$	$[1,1]$	$2^1 \cdot 1^1$
$false$	$false$	$[2,0]$	$2^2 \cdot 1^0$
⋮	⋮	⋮	⋮

~~Grounding?~~

- Logvar D in g_1 grounded $= \phi(Epid, Man(W), Nat(d_1), Nat(d_2))$
  - 16 mappings

$Epid$	$Man(W)$	$Nat(D)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

$Epid$	$Man(W)$	$Nat(d_1)$	$Nat(d_2)$	ϕ
$false$	$false$	$false$	$false$	$1 \cdot 1$
$false$	$false$	$false$	$true$	$1 \cdot 2$
$false$	$false$	$true$	$false$	$2 \cdot 1$
$false$	$false$	$true$	$true$	$2 \cdot 2$
$false$	$true$	$false$	$false$	$3 \cdot 3$
$false$	$true$	$false$	$true$	$3 \cdot 4$
$false$	$true$	$true$	$false$	$4 \cdot 3$
$false$	$true$	$true$	$true$	$4 \cdot 4$
$true$	$false$	$false$	$false$	$5 \cdot 5$
$true$	$false$	$false$	$true$	$5 \cdot 6$
$true$	$false$	$true$	$false$	$6 \cdot 5$
$true$	$false$	$true$	$true$	$6 \cdot 6$
$true$	$true$	$false$	$false$	$7 \cdot 7$
$true$	$true$	$false$	$true$	$7 \cdot 8$
$true$	$true$	$true$	$false$	$8 \cdot 7$
$true$	$true$	$true$	$true$	$8 \cdot 8$

Counting!

- Logvar D in g_1 counted $= \phi_1'(Epid, Man(W), \#_D[Nat(D)])$
  - 12 mappings

$Epid$	$Man(W)$	$Nat(D)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

$Epid$	$Man(W)$	$\#_D[Nat(D)]$	ϕ_1'
$false$	$false$	[0,2]	$2^0 \cdot 1^2$
$false$	$false$	[1,1]	$2^1 \cdot 1^1$
$false$	$false$	[2,0]	$2^2 \cdot 1^0$
$false$	$true$	[0,2]	$4^0 \cdot 3^2$
$false$	$true$	[1,1]	$4^1 \cdot 3^1$
$false$	$true$	[2,0]	$4^2 \cdot 3^0$
$true$	$false$	[0,2]	$6^0 \cdot 5^2$
$true$	$false$	[1,1]	$6^1 \cdot 5^1$
$true$	$false$	[2,0]	$6^2 \cdot 5^0$
$true$	$true$	[0,2]	$8^0 \cdot 7^2$
$true$	$true$	[1,1]	$8^1 \cdot 7^1$
$true$	$true$	[2,0]	$8^2 \cdot 7^0$

Count Conversion

- Remember: Counting a logvar binds a logvar, i.e., removes logvar from the logvars of the parfactor
  - E.g.,
    - $g_1 = \phi_1(Epid, Man(W), Nat(D)) \rightarrow lv(g_1) = \{D, W\}$
    - $g_1' = \phi_1'(Epid, Man(W), \#_D[Nat(D)]) \rightarrow lv(g_1') = \{W\}$
  - Helps with Precondition 2 of summing out!
    - Precondition 2: PRV to sum out has to contain all logvars of parfactor

- Operator $\mathrm{count{-}convert}$
  - Count a logvar → convert a PRV into a (P)CRV
  - Works as an ”enabler” for $\mathrm{sum{-}out}$ operator
  - Preconditions for $\mathrm{count{-}convert}$ as well

Preconditions

- Inputs: Logvar X occurring in parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
- Three preconditions for correctness
  1. There is exactly one PRV $A_i \in rv(g)$ s.t. $X \in lv(A)$
     - For this simple version of count conversion
     - More generalised versions exist to count

Preconditions

- Inputs: Logvar X occurring in parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
- Three preconditions for correctness
  2. Logvar X is count-normalised w.r.t. $\boldsymbol{X} \setminus \{X\}$ in C, with $\boldsymbol{X}$ the set of $\mathcal{X}$
     - Identical histograms for all cases of $\boldsymbol{X} \setminus \{X\}$

Each grounding of $\boldsymbol{X} \setminus \{X\}$ needs have the same number of groundings of X in C ($= Y$ in graphics)

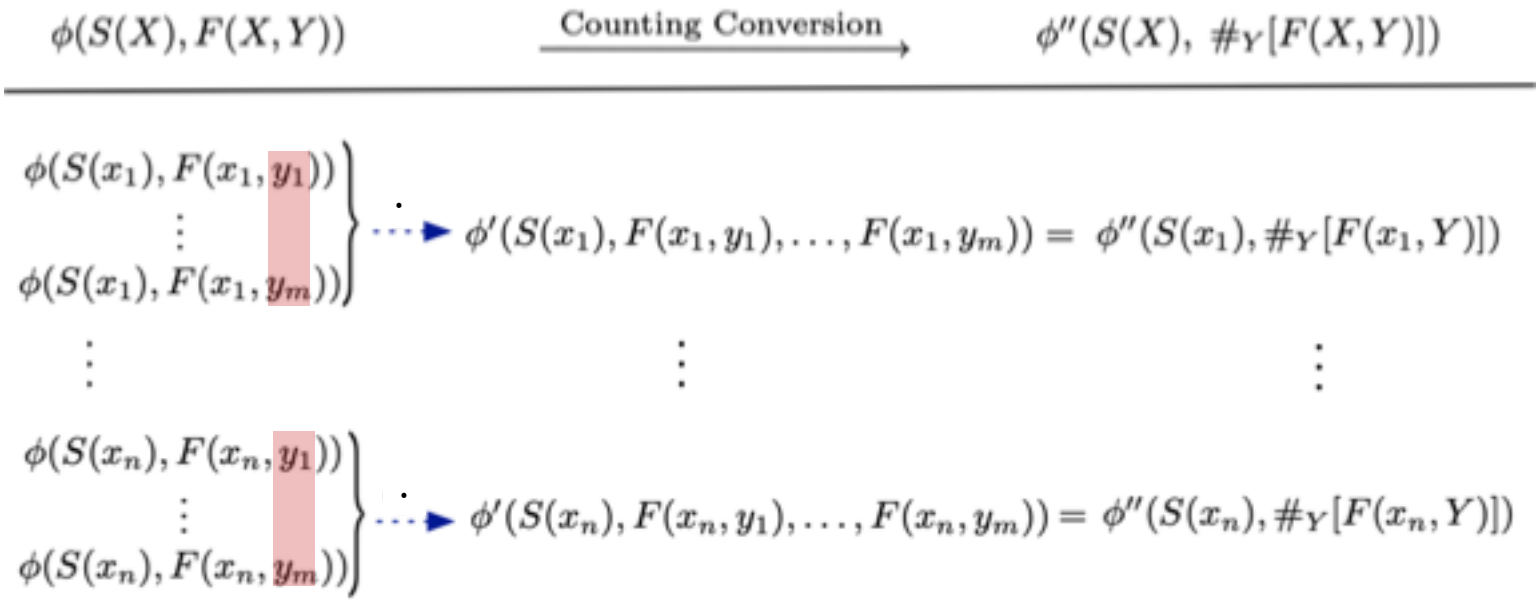


Figure taken from: Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013.

Preconditions

- Inputs: Logvar X occurring in parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
- Three preconditions for correctness
  3. For all counted logvars $X^{\#}$ in g:
     $$\pi_{X,X^{\#}}(C_{\mathcal{X}}) = \pi_X(C_{\mathcal{X}}) \times \pi_{X^{\#}}(C_{\mathcal{X}})$$
     - I.e., no inequality constraint exists between X and any other counted logvar in g
     - Test it out
       - with
         $\phi(R(X), \#_Y[S(Y)])_{|((X,Y),\{(x_1,y_2),(x_1,y_3),(x_2,y_1),(x_2,y_3),(x_3,y_1),(x_3,y_2)\})}$
         - Inequality constraint $X \neq Y$ encoded in constraint
         - Does not fulfil $\pi_X(C_{\mathcal{X}}) \times \pi_{X^{\#}}(C_{\mathcal{X}})$
       - and
         $\phi(R(X), \#_Y[S(Y)])_{|((X,Y),\{x_1,x_2,x_3\}\times\{y_1,y_2,y_3\})}$
         - Fulfils $\pi_X(C_{\mathcal{X}}) \times \pi_{X^{\#}}(C_{\mathcal{X}})$
       - By comparing full joints before and after counting X

Count Conversion: Operator

- Inputs:
  - Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
  - Logvar X occurring in $\mathcal{X}$ for counting
- Preconditions:
  - There is exactly one PRV $A_i \in rv(g)$ s.t. $X \in lv(A)$
  - X is count-normalised w.r.t. $\boldsymbol{X} \setminus \{X\}$ in C, with $\boldsymbol{X}$ the set of $\mathcal{X}$
  - For all counted logvars $X^{\#}$ in g: $\pi_{X,X^{\#}}(C_{\mathcal{X}}) = \pi_X(C_{\mathcal{X}}) \times \pi_{X^{\#}}(C_{\mathcal{X}})$
- Output: $\phi'(\mathcal{A}')_{|C}$
  - $\mathcal{A}' = (A_1, \dots, A_{i-1}) \circ (A_i') \circ (A_{i+1}, \dots, A_n), A_i' = \#_X[A_i]$
  - For each assignment $\boldsymbol{a}' = (\dots, a_{i-1}, h, a_{i+1}, \dots)$ to $\mathcal{A}'$,

$$\phi'(\dots, a_{i-1}, h, a_{i+1}, \dots) = \prod_{a_i \in \mathcal{R}(A_i)} \phi(\dots, a_{i-1}, a_i, a_{i+1}, \dots)^{h(a_i)}$$

    - With $h(a_i)$ denoting the count of a_i in histogram h
- Postcondition:

$$G \sim G \setminus \{g\} \cup \{\text{count-convert}(g, X)\}$$

Count Conversion: Example Revisited

- From

$$\phi_1(Epid, Man(W), Nat(D))$$

- To

$$\phi_1'(Epid, Man(W), \#_D[Nat(D)])$$

$Epid$	$Man(W)$	$Nat(D)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

- D occurs only in $Nat(D)$
- D is count-normalised w.r.t. W in $\big((W, D), \mathcal{D}(W) \times \mathcal{D}(D)\big)$
- No other counted logvar

Count Conversion: Example Revisited

- Converting $Nat(D)$ into $\#_D[Nat(D)]$

$$\phi'(\dots, a_{i-1}, h, a_{i+1}, \dots) = \prod_{a_i \in \mathcal{R}(A_i)} \phi(\dots, a_{i-1}, a_i, a_{i+1}, \dots)^{h(a_i)}$$

$Epid$	$Man(W)$	$Nat(D)$	ϕ_1
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	3
$false$	$true$	$true$	4
$true$	$false$	$false$	5
$true$	$false$	$true$	6
$true$	$true$	$false$	7
$true$	$true$	$true$	8

$Epid$	$Man(W)$	$\#_D[Nat(D)]$	ϕ'_1
$false$	$false$	$[0,2]$	$2^0 \cdot 1^2$
$false$	$false$	$[1,1]$	$2^1 \cdot 1^1$
$false$	$false$	$[2,0]$	$2^2 \cdot 1^0$
$false$	$true$	$[0,2]$	$4^0 \cdot 3^2$
$false$	$true$	$[1,1]$	$4^1 \cdot 3^1$
$false$	$true$	$[2,0]$	$4^2 \cdot 3^0$
$true$	$false$	$[0,2]$	$6^0 \cdot 5^2$
$true$	$false$	$[1,1]$	$6^1 \cdot 5^1$
$true$	$false$	$[2,0]$	$6^2 \cdot 5^0$
$true$	$true$	$[0,2]$	$8^0 \cdot 7^2$
$true$	$true$	$[1,1]$	$8^1 \cdot 7^1$
$true$	$true$	$[2,0]$	$8^2 \cdot 7^0$

Generalised Counting

- Count conversion as discussed here, first introduced by Milch et al. (2008)
- Generalised counting by Nima Taghipour et al. (2013)
  1. Count logvars that appear in more than one PRV
     - E.g., $\phi\big(Q(X), R(X), S(Y), T(Y)\big)$ $\rightarrow \phi\big(\#_X[Q(X), R(X)], S(Y), T(Y)\big)$
  2. Merge CRVs with counted logvars of the same domain
     - E.g., $\phi(\#_X[Q(X), R(X)])_{C^X}$ and $\phi(\#_Y[Q(Y), R(Y)])_{|C^Y}$ with $gr\big(X_{|C^X}\big) = gr\big(Y_{|C^Y}\big)$ $\rightarrow \phi(\#_X[Q(X), R(X)])_C$
  3. Merge-count a PRV and a CRV with an inequality constraint
     - E.g., $\phi\big(\#_X[Q(X)], R(Y)\big)_C$ with C encoding $X \neq Y$ $\rightarrow \phi(\#_X[Q(X), R(X)]\,)_C$

Brian Milch, Luke S. Zettelmoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling: Lifted Probabilistic Inference with Counting Formulas. In: *AAAI-08 Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.
Nima Taghipour and Jesse Davis: Generalised Counting for Lifted Variable Elimination. In: *ILP-13 Proceedings of the International Conference on Inductive Logic Programming*, 2013. (or in Nima Taghipour's PhD thesis)

Splitting

- Precondition 1 of $\mathrm{sum}-\mathrm{out}$ operator: PRV A under $(\mathcal{X}, C_{\mathcal{X}})$ only occurs in g
- Formalism is very flexible in terms of constraints
  - E.g., $\phi_1\big(R(X)\big)_{(X,\{x_1,x_2,x_3\})}$ vs. $\phi_2\big(R(X)\big)_{(X,\{x_1,x_2,x_3,x_4x_5\})}$
- Split parfactor s.t. the set of constants occurring in constraints for a logvar are

  either *identical* or *disjoint*

  - I.e., no overlaps between sets of constants per logvar
  - E.g., split $\phi_2\big(R(X)\big)_{(X,\{x_1,x_2,x_3,x_4x_5\})}$ into
    - $\phi_2\big(R(X)\big)_{(X,\{x_1,x_2,x_3\})}$
    - $\phi_2\big(R(X)\big)_{(X,\{x_4x_5\})}$

Splitting will become important for evidence handling and how we handle query terms

Splitting on Overlap

- Splitting a constraint $C_1 = (\mathcal{X}_1, C_{\mathcal{X}_1})$ on its $\boldsymbol{Y}$-overlap with a constraint $C_2 = (\mathcal{X}_2, C_{\mathcal{X}_2})$, denoted $C_1/_{\boldsymbol{Y}}C_2$, partitions $C_{\mathcal{X}_1}$ into two subsets containing all tuples for which the $\boldsymbol{Y}$ part occurs or does not occur, respectively

$$C_1/_{\boldsymbol{Y}}C_2 = \left\{ \begin{pmatrix} (\mathcal{X}_1), \{t \in C_{\mathcal{X}_1} \mid \pi_{\boldsymbol{Y}}(\{t\}) \in \pi_{\boldsymbol{Y}}(C_{\mathcal{X}_2})\} \end{pmatrix} \\ \begin{pmatrix} (\mathcal{X}_1), \{t \in C_{\mathcal{X}_1} \mid \pi_{\boldsymbol{Y}}(\{t\}) \notin \pi_{\boldsymbol{Y}}(C_{\mathcal{X}_2})\} \end{pmatrix} \right\}$$

- Parfactor partitioning
  Given a parfactor $g = \phi(\mathcal{A})_{|C}$ and a partition $\mathbb{C} = \{C_i\}_{i=1}^n$ of C,

$$\text{partition}(g, \mathbb{C}) = \left\{\phi(\mathcal{A})_{|C_i}\right\}_{i=1}^n$$

Splitting on Overlap: Example

- Consider $\phi\big(R(X), T(X,Y)\big)_{|C_1}$
  - $C_1 = \big((X,Y), \{x_1, x_2, x_3, x_4, x_5\} \times \{y_1, y_2\}\big)$
  - $C_2 = \big((X), \{x_1, x_2, x_3\}\big)$
- Splitting C_1 on its $\boldsymbol{Y} = \{X\}$-overlap with C_2

$$C_1/_{\boldsymbol{Y}}C_2 = \left\{ \begin{array}{l} \Big((X,Y), \big\{t \in C_{(X,Y)} \mid \pi_X(\{t\}) \in \{x_1, x_2, x_3\}\big\}\Big) \\ \Big((X,Y), \big\{t \in C_{(X,Y)} \mid \pi_X(\{t\}) \notin \{x_1, x_2, x_3\}\big\}\Big) \end{array} \right\}$$

$$= \left\{ \begin{array}{c} \big((X,Y), \{x_1, x_2, x_3\} \times \{y_1, y_2\}\big) \\ \big((X,Y), \{x_4, x_5\} \times \{y_1, y_2\}\big) \end{array} \right\}$$

- Partitioning

$$\text{partition}(g, C_1/_{\boldsymbol{Y}}C_2) = \left\{ \begin{array}{l} \phi\big(R(X), T(X,Y)\big)_{|((X,Y), \{x_1, x_2, x_3\} \times \{y_1, y_2\})} \\ \phi\big(R(X), T(X,Y)\big)_{|((X,Y), \{x_4, x_5\} \times \{y_1, y_2\})} \end{array} \right\}$$

Splitting: Operator

- Inputs:
  - Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
  - PRV $A = R(\boldsymbol{Y})$ occurring in $\mathcal{A}$
  - PRV $A' = R(\boldsymbol{Y})_{|C'}$ or $\#_Y\left[R(\boldsymbol{Y})_{|C'}\right]$
- Output:

$$\text{partition}(g, \mathbb{C}), \mathbb{C} = C/_{\boldsymbol{Y}}C' \setminus \emptyset$$

- Postcondition:

$$G \sim G \setminus \{g\} \cup \text{split}(g, A, A')$$

Splitting: Example

- Inputs:
  - Parfactor $\phi\big(R(X), T(X,Y)\big)_{|C_1}$
    - $C_1 = \big((X,Y), \{x_1, x_2, x_3, x_4, x_5\} \times \{y_1, y_2\}\big)$
  - $R(X)$
  - $R(X)_{|C_2}, C_2 = \big((X), \{x_1, x_2, x_3\}\big)$
- Output:

$$\text{partition}(g, C_1 /_{\boldsymbol{Y}} C_2)$$

$$= \left\{ \begin{array}{l} \phi\big(R(X), T(X,Y)\big)_{|((X,Y), \{x_1, x_2, x_3\} \times \{y_1, y_2\})} \\ \phi\big(R(X), T(X,Y)\big)_{|((X,Y), \{x_4, x_5\} \times \{y_1, y_2\})} \end{array} \right\}$$

Other Operators

- Further "enablers" of lifted summing out all are *variants of splitting on overlap* and *partitioning*
- Splitting of CRVs
  - More complex as histograms have to be split
    - E.g., a histogram $[1,3]$ for $\{x_1, x_2, x_3, x_4\}$ may have to be split on $\{x_1, x_2\}$
  - Operator called expand
- Count-normalisation
  - Split a constraint s.t. in the set of resulting constraints, each constraint is count-normalised w.r.t. to desired $\boldsymbol{Y}_i|\boldsymbol{Z}_i$ property
    - Group sets of constants by the different counts $\mathrm{ncount}_{\boldsymbol{Y}_i|\boldsymbol{Z}_i}(C_i)$ they yield
  - Operator called count−normalise
- Grounding: the last resort
  - Splitting on individual constants
  - Operator ground as expected
- More information:

  Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013. (or in Nima Taghipour's PhD thesis)

Lifted Absorption

- Remember:
- Observations for groundings of a PRV can be
  - One of the range values
  - Not available (N/A)
- Compactly encode evidence with PRVs and parfactors
  - E.g., for $Sick(X)$ with $\mathcal{D}(X) = \{x_1, \dots, x_n\}$
    - $Sick(x_1) = Sick(x_2) = \cdots = Sick(x_{10}) = true$
      - Parfactor $g_e^T = \phi_e^T\big(Sick(X)\big)_{((X),\{x_1,\dots,x_{10}\})}$
    - $Sick(x_{11}) = Sick(x_{12}) = \cdots = Sick(x_{20}) = false$
      - Parfactor $g_e^F = \phi_e^F\big(Sick(X)\big)_{((X),\{x_{11},\dots,x_{20}\})}$
    - Observations for $Sick(x_{21})$ … $Sick(x_n)$ N/A
  - Within each group:
    instances are indistinguishable again

$Sick(X)$	ϕ_e^T
$false$	0
$true$	1

$Sick(X)$	ϕ_e^F
$false$	1
$true$	0

→ Absorb evidence for each group at once using the parfactors

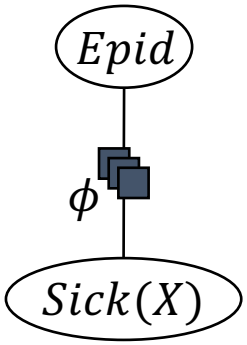
Splitting based on Evidence

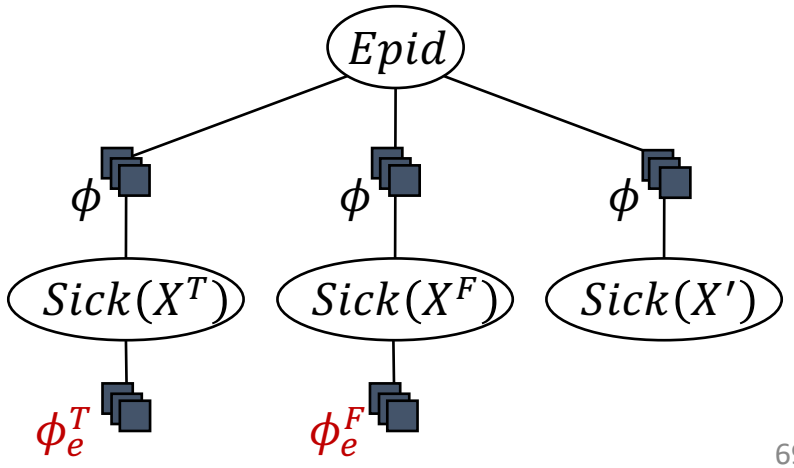
- As observations are seldom for all constants in a constraint, parfactors have to be split based on the constants that occur in the observations
  - Called shattering on evidence
  - Only then:
    absorb applicable evidence in each parfactor individually
  - E.g., given evidence parfactors
    - $g_e^T = \phi_e^T\big(Sick(X)\big)_{|((X),\{x_1,\ldots,x_{10}\})}$
    - $g_e^F = \phi_e^F\big(Sick(X)\big)_{|((X),\{x_{11},\ldots,x_{20}\})}$

    every parfactor containing $Sick(X)$ has to be split on the constraints
    - For readability, one can introduce new logvars with domains as given in the constraints of the evidence parfactors to distinguish the logvars in the parfactor graph

Splitting based on Evidence: Example

- X being renamed
  - X^T for constants $\{x_1, \ldots, x_{10}\}$,
  - X^F for constants $\{x_{11}, \ldots, x_{20}\}$, and
  - X' for the remaining constants $\{x_{21}, \ldots, x_n\}$
- yields evidence parfactors
  - $g_e^T = \phi_e^T\big(Sick(X)\big)_{|((X),\{x_1,\ldots,x_{10}\})} = \phi_e^T\big(Sick(X^T)\big)_{|\mathsf{T}}$
  - $g_e^F = \phi_e^F\big(Sick(X)\big)_{|((X),\{x_{11},\ldots,x_{20}\})} = \phi_e^F\big(Sick(X^F)\big)_{|\mathsf{T}}$
- The model on the left is shattered on g_e^T, g_e^F, i.e., split on the constraints in g_e^T, g_e^F, yielding the model on the right
  - Evidence parfactors can be drawn as well





Shattering on Evidence

- Given a set of evidence parfactors $\{g_e\}_{e=1}^{m}$ and a model $G = \{g_i\}_{i=1}^{n}$
- For each $g_e = \phi_e\big(R(\boldsymbol{X})\big)_{|C_e}$:
  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $R(\boldsymbol{X}) \in rv(g_i)$:
      - Split g_i on C, i.e.,

        $G \leftarrow G \setminus \{g_i\} \cup \text{split}\big(g_i, R(\boldsymbol{X}), R(\boldsymbol{X})_{|C_e}\big)$

        - Output of split: $\text{partition}(g_i, \mathbb{C}), \mathbb{C} = C_i/_{\boldsymbol{X}} C_e \setminus \emptyset$

- After shattering, absorb each evidence parfactor g_e in each applicable parfactor g_i
  - Possible to interleave shattering and absorption

Evidence Absorption

- Given a set of evidence parfactors $\{g_e\}_{e=1}^{m}$ and a model $G = \{g_i\}_{i=1}^{n}$
- For each $g_e = \phi_e\big(R(\boldsymbol{X})\big)_{|C_e}$:
  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $R(\boldsymbol{X}) \in rv(g_i)$:
      - Absorb g_e in g_i
- Informally, absorb evidence means
  - Analogous to propositional absorption
  - Set values to 0 where range value ≠ observation
    - Equivalent to multiplying g with g_e
  - Eliminate variable
    - Drop lines with values set to 0
    - Drop column of evidence PRV

Evidence Absorption: Example

- Observations as before
  - $Sick(X^T) = true$ in g_e^T
  - $Sick(X^F) = false$ in g_e^F

$Epid$	$Sick(X^T)$	ϕ^T
$false$	$false$	~~5~~ 0
$false$	$true$	1
$true$	$false$	~~4~~ 0
$true$	$true$	6

$Epid$	$Sick(X^F)$	ϕ^F
$false$	$false$	5
$false$	$true$	~~1~~ 0
$true$	$false$	4
$true$	$true$	~~6~~ 0

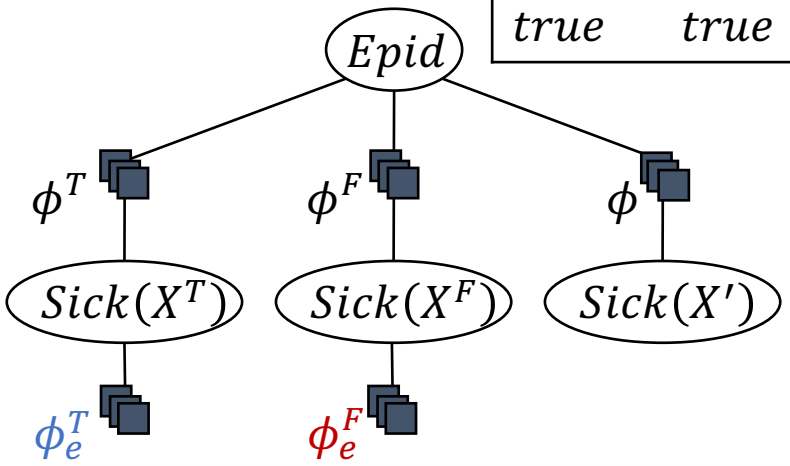
$Epid$	$Sick(X^T)$	ϕ^T
$false$	$true$	1
$true$	$true$	6

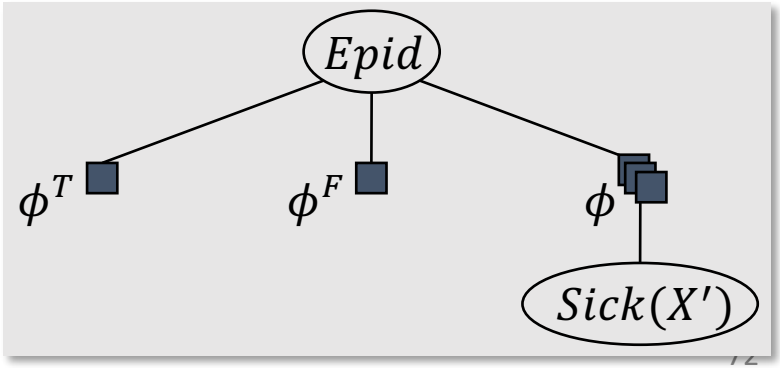
$Epid$	$Sick(X^F)$	ϕ^F
$false$	$false$	5
$true$	$false$	4

$Epid$	ϕ^T
$false$	1
$true$	6

$Epid$	ϕ^F
$false$	1
$true$	6

$Epid$	$Sick(X)$	ϕ
$false$	$false$	5
$false$	$true$	1
$true$	$false$	4
$true$	$true$	6





Absorption as Multiplication

- Multiply evidence parfactor g_e into model parfactor g_i if $rv(g_e) \subseteq rv(g_i)$
  - After shattering

$Epid$	$Sick(X^T)$	ϕ
$false$	$false$	5
$false$	$true$	1
$true$	$false$	4
$true$	$true$	6

$\cdot$

$Sick(X^T)$	ϕ_e^T
$false$	0
$true$	1

$=$

$Epid$	$Sick(X^T)$	ϕ^T
$false$	$false$	$5 \cdot 0$
$false$	$true$	$1 \cdot 1$
$true$	$false$	$4 \cdot 0$
$true$	$true$	$6 \cdot 1$

- Then, drop lines with zeroes and eliminate column

$=$

$Epid$	$Sick(X^T)$	ϕ^T
$false$	$false$	1
$true$	$false$	6

$=$

$Epid$	ϕ^T
$false$	1
$true$	6

Preconditions

- Inputs: Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$ with PRV $A_i = R(\boldsymbol{Y})$ and evidence $g_e = \phi\big(R(\boldsymbol{Y})\big)_{|C_e}$
- Preconditions:
  - $gr\big(A_{i|C}\big) \subseteq gr\big(A_{i|C_e}\big)$
    - All groundings represented by $A_{i|C}$ have the same evidence as encoded in g_e
      - Treat as indistinguishable
  - Not-counted logvars exclusive to A_i, $\boldsymbol{X}^{nce}$, are count-normalised w.r.t. to the remaining not-counted logvars, $\boldsymbol{X}^{ncr}$, in C, i.e., $r = \mathrm{ncount}_{\boldsymbol{X}^{nce}|\boldsymbol{X}^{ncr}}(C)$ exists
    - Each remaining sequence of constants references the same number of constants that are eliminated

Lifted Absorption: Operator

- Inputs:
  - Parfactor $g = \phi(\mathcal{A})_{|C}, C = (\mathcal{X}, C_{\mathcal{X}})$
  - PRV $A_i = R(\boldsymbol{Y})$ or (P)CRV $A_i = \#_X[R(\boldsymbol{Y})]$ occurring in $\mathcal{A}$
  - Evidence parfactor $g_e = \phi\big(R(\boldsymbol{Y})\big)_{|C_e}$ with $o =$ observed value of $R(\boldsymbol{Y})$ in g_e
- Let
  - $\boldsymbol{X}^{excl} = \boldsymbol{Y} \setminus lv(\boldsymbol{A} \setminus \{A_i\})$ (exclusive logvars of A_i)
  - $\boldsymbol{X}^{nce} = lv(A_i) \setminus lv(\boldsymbol{A} \setminus \{A_i\})$ (not-counted exclusive logvars of A_i)
  - $\boldsymbol{X}^{rem} = lv(\mathcal{X}) \setminus \boldsymbol{X}^{excl}$ (logvars remaining in g)
  - $\boldsymbol{X}^{ncr} = lv(\mathcal{A}) \setminus \boldsymbol{X}^{excl}$ (not-counted logvars remaining in g)
- Preconditions:
  - $gr\big(A_{i|C}\big) \subseteq gr\big(A_{i|C_e}\big)$
  - $\boldsymbol{X}^{nce}$ is count-normalised w.r.t. $\boldsymbol{X}^{ncr}$ in C, i.e., $r = \mathrm{ncount}_{\boldsymbol{X}^{nce}|\boldsymbol{X}^{ncr}}(C)$ exists
- Output: $g' = \phi'(\mathcal{A}')_{|C'}, C' = (\mathcal{X}', C_{\mathcal{X}'})$
  - $\mathcal{A}' = (A_1, \ldots, A_{i-1}) \circ (A_{i+1}, \ldots, A_n)$
  - $\mathcal{X}' = \pi_{\boldsymbol{X}^{rem}}(\mathcal{X})$
  - $C_{\mathcal{X}'} = \pi_{\boldsymbol{X}^{rem}}(C_{\mathcal{X}})$
  - $\phi'(\ldots, a_{i-1}, a_{i+1}, \ldots) = \phi(\ldots, a_{i-1}, e, a_{i+1}, \ldots)^r$
    - with $e = o$ if $A_i = R(\boldsymbol{Y})$ and
    - otherwise $e =$ a histogram with $e(o) = \mathrm{ncount}_{X|lv(\mathcal{A})}(C)$ and $e(o') = 0, o' \neq o$
- Postcondition:

$$G \cup \{g_e\} \sim G \setminus \{g\} \cup \{g_e, \mathrm{absorb}(g, A_i, g_e)\}$$

Lifted Absorption: Example Revisited

- Output: $g' = \phi'(\mathcal{A}')_{|C'}$
  - $\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, e, a_{i+1}, \dots)^r$
    - with $e = o$ if $A_i = R(\boldsymbol{Y})$
- E.g.,
  - in g_e^T, $o = true$
  - $g = \phi\big(Epid, Sick(X^T)\big)$
  - Output: $g' = \phi'(Epid)$

$Sick(X^T)$	ϕ_e^T
$false$	0
$true$	1

$Epid$	ϕ'
$false$	1
$true$	6

$Epid$	$Sick(X^T)$	ϕ
$false$	$false$	5
$false$	$true$	1
$true$	$false$	4
$true$	$true$	6

- Actually: If observation given as histogram, absorption works as with PRVs
  - E.g.
    - Observing one time $Epid(e_i) = true$ and three times $Epid(e_i) = false$
    - *Careful*: Does not specify which constants observed with which value
  - Normally observations are for groundings even if they appear in a CRV

$\#_E[Epid(E)]$	ϕ_e^T
[0,4]	0
[1,3]	1
[2,2]	0
[3,1]	0
[4,0]	0

Lifted Absorption: Evidence for CRVs

- Output: $g' = \phi'(\mathcal{A}')_{|C'}$
  - $\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, e, a_{i+1}, \dots)^r$
    - $e =$ a histogram with $e(o) = \mathrm{ncount}_{X|lv(\mathcal{A})}(C)$ and $e(o') = 0$, $o' \neq o$
- Evidence PRV appears as inner PRV of a (P)CRV
  - Turn observations into histogram
    - All groundings have the same observation
    - Peak-shaped histogram with $\mathrm{ncount}_{X|lv(\mathcal{A})}(C)$ at position o and 0 otherwise
  - E.g.,
    - observations $Nat(D) = false$ for all $gr(Nat(D)) \rightarrow o = false$ in g_e
    - Parfactor: $g = \phi(Epid, \#_D[Nat(D)])$
      - $\mathrm{ncount}_{D|\emptyset}(\mathsf{T}) = 2$
      - Forms histogram: [0,2]
    - Output: $g' = \phi'(Epid)$

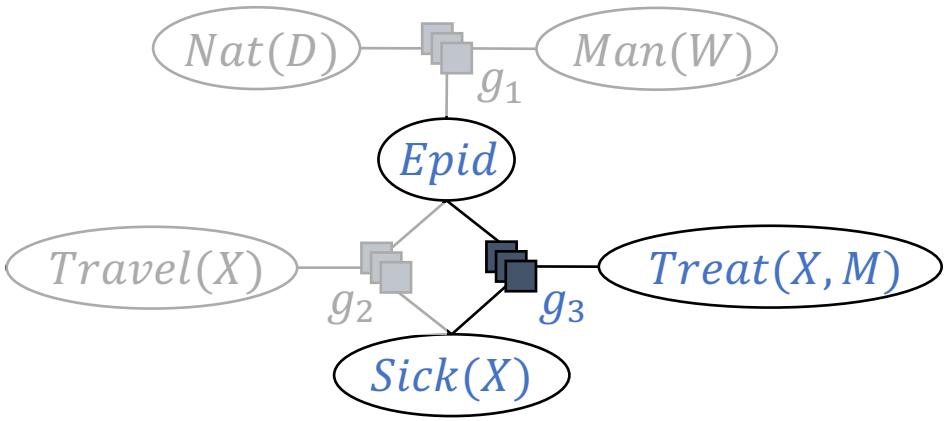
$Nat(D)$	ϕ_e
$false$	1
$true$	0

$Epid$	$\#_D[Nat(D)]$	$\phi^{\#}$
$false$	[0,2]	1
$false$	[1,1]	2
$false$	[2,0]	3
$true$	[0,2]	4
$true$	[1,1]	5
$true$	[2,0]	6

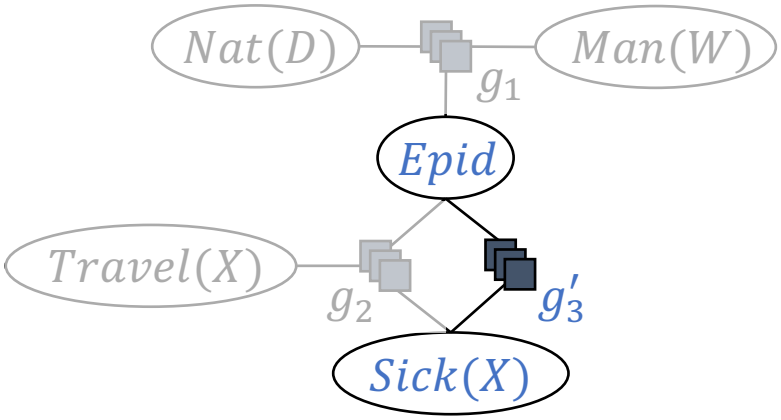
$Epid$	ϕ_e
$false$	1
$true$	4

Lifted Absorption: Eliminating a Logvar

- Output: $g' = \phi'(\mathcal{A}')_{|C'}$
  - $\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, e, a_{i+1}, \dots)^r$
    - with $e = o$ if $A_i = R(\boldsymbol{Y})$
- E.g., observations $Treat(X, M) = true$ $\forall (x, m) \in \mathsf{T}$
  - Parfactor g_3 contains $Treat(X, M)$
  - Output: $\phi'\big(Epid, Sick(X)\big)$
    - Absorbing $Treat(X, M) = true$ eliminates M
    - $r = \text{ncount}_{M|X}(C) = 2$
    - Potentials in selected lines have to be raised to the power of 2



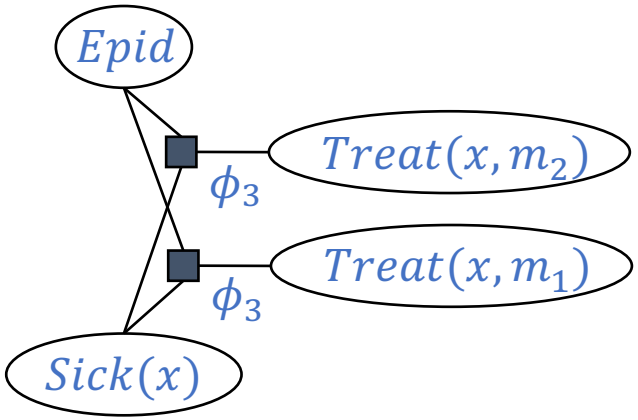
Initial model



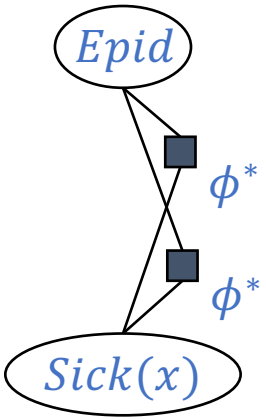
After lifted absorb

Lifted Absorption: Eliminating a Logvar

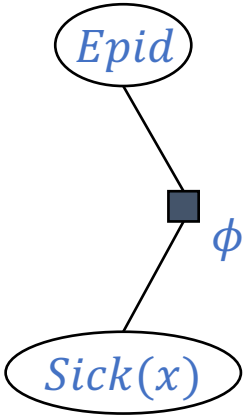
- Output: $g' = \phi'(\mathcal{A}')_{|C'}$
  - $\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, e, a_{i+1}, \dots)^r$
    - with $e = o$ if $A_i = R(\boldsymbol{Y})$
- E.g., observations $Treat(X, M) = true \; \forall (x, m) \in \mathsf{T}$
  - Equivalent ground case:
    - Absorb $Treat(x, m) = true$ in r propositional factors for each x
    - Output: $\phi^*\big(Epid, Sick(x)\big)$ r times for each x
    - Multiply all $\phi^*\big(Epid, Sick(x)\big)$ into one factor $\phi'\big(Epid, Sick(x)\big)$, i.e., raise to the power of r



Initial model



After absorb



After multiply

Shattering on Evidence & Absorption

- Given a set of evidence parfactors $\{g_e\}_{e=1}^{m}$ and a model $G = \{g_i\}_{i=1}^{n}$
- For each $g_e = \phi_e\big(R(\boldsymbol{X})\big)_{|C_e}$:

  Shattering

  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $R(\boldsymbol{X}) \in rv(g_i)$:
      - Split g_i on C, i.e.,

        $G \leftarrow G \setminus \{g_i\} \cup \text{split}\big(g_i, R(\boldsymbol{X}), R(\boldsymbol{X})_{|C_e}\big)$
- For each $g_e = \phi_e\big(R(\boldsymbol{X})\big)_{|C_e}$:

  Absorption

  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $R(\boldsymbol{X}) \in rv(g_i)$:
      - Absorb g_e in g_i

        $G \leftarrow G \setminus \{g_i\} \cup \text{absorb}(g_i, R(\boldsymbol{X}), g_e)$

Shattering

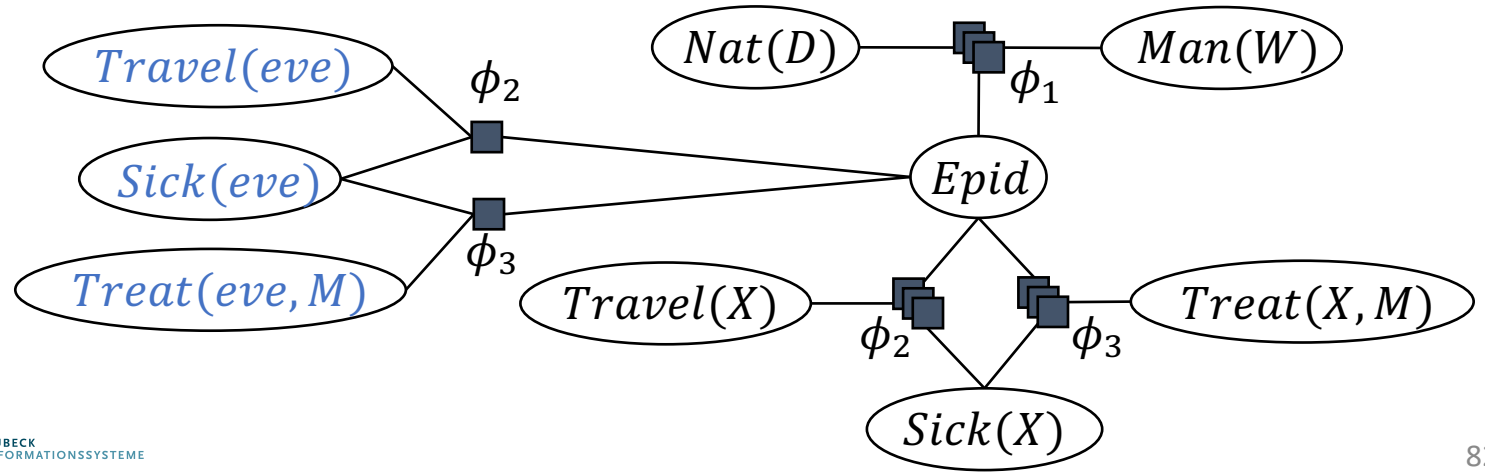
- Shattering on *evidence*
  - Splitting on evidence
  - To allow for absorbing evidence in groups
    - Constants of different groups distinguishable
    - Constants within each group indistinguishable
- Shattering on *query terms*
  - Splitting off the groundings appearing in queries
  - To allow for eliminating all non-query terms
    - Query terms are different, as they have to remain
      → query terms not indistinguishable

Shattering on Query Terms

- Given a set of ground query terms $\boldsymbol{S}$ from a query $P(\boldsymbol{S}|\boldsymbol{T})$ and a model $G = \{g_i\}_{i=1}^{n}$
- For each $R(\boldsymbol{x}) \in \boldsymbol{S}$ with $\boldsymbol{x} \neq ()$:
  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $R(\boldsymbol{X}) \in rv(g_i)$:
      - Split g_i on $\boldsymbol{x}$, i.e.,

$$G \leftarrow G \setminus \{g_i\} \cup \text{split}\big(g_i, R(\boldsymbol{X}), R(\boldsymbol{X})_{|(\mathcal{X},\{\boldsymbol{x}\})}\big)$$

- E.g.,
  - $P(Epid)$ → no constants, no splitting
  - $P\big(Sick(eve)\big)$ → splitting off $Sick(eve)$



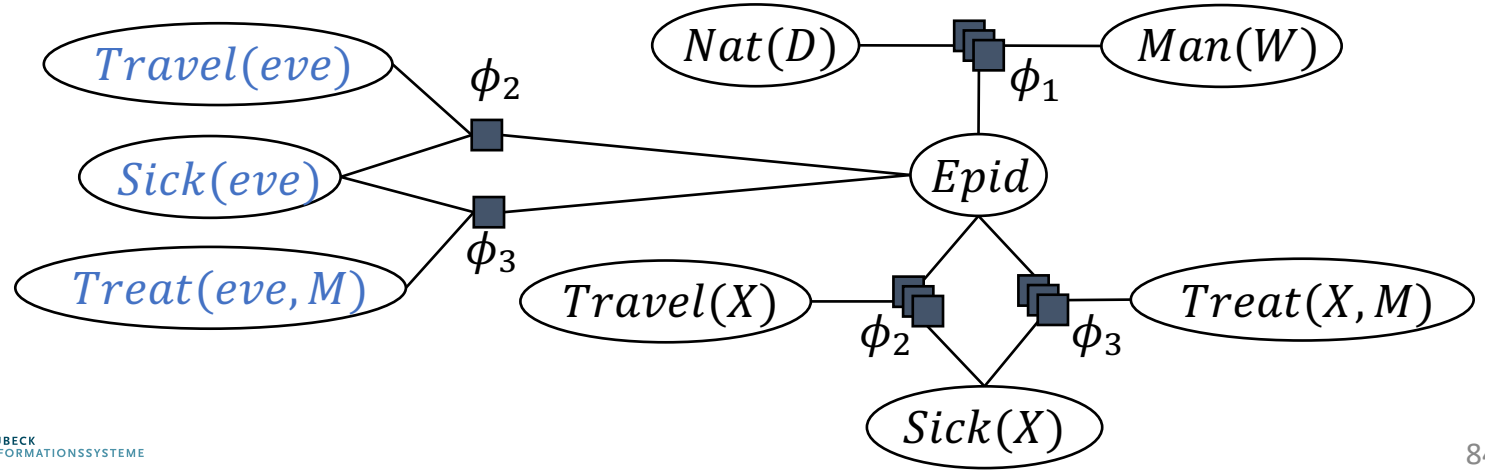
Types of Shattering

- Pre-emptive shattering
  - Recursively shattering the model on evidence, query terms, and itself before starting with any calculations
    - Shattering a model on *itself*:
      - Ensure that all sets of constants for logvars occurring in constraints are either identical or disjoint
        - Allows for introducing one logvar for each set of constants and $\top$ constraints in parfactors except when an inequality is encoded
        - Avoids splitting during an LVE run and makes PRV comparisons easier
    - If the input model is already shattered, the if-condition in shattering on evidence and query terms changes to
      - $\boldsymbol{X} \cap lv(g_i) \neq \emptyset$
        - Instead of $R(\boldsymbol{X}) \in rv(g_i)$
        - Split if any of the logvars occur, not only if the PRV occurs

One can come up with other variants like $\boldsymbol{X} \supseteq lv(g_i)$

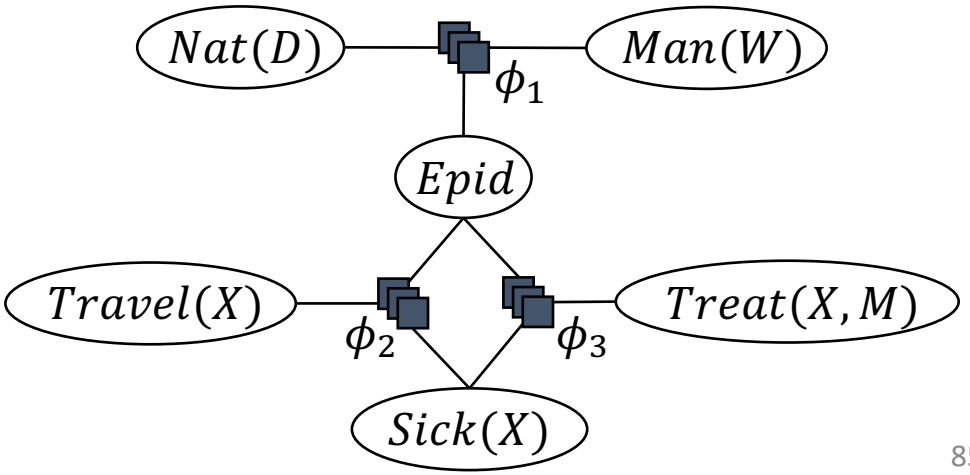
Pre-emptive Shattering: Example

- $P\big(Travel(eve)\big) \rightarrow$ splitting off eve
  - Splits g_2 and g_3 instead of only g_2, $\mathcal{D}(X) = \mathcal{D}(X) \setminus \{eve\}$
- $P\big(Treat(eve, m_2)\big) \rightarrow$ splitting off eve and m_2
  - Splits g_2 as with $P\big(Travel(eve)\big)$
  - Splits g_3 on eve and m_2
    - With $\mathcal{D}(X) = \mathcal{D}(X) \setminus \{eve\}$ and $\mathcal{D}(M) = \mathcal{D}(M) \setminus \{m_2\}$
    - $g_3 = \phi_3\big(Epid, Sick(X), Treat(X, M)\big)$
    - $g_3' = \phi_3\big(Epid, Sick(eve), Treat(eve, M)\big)$
    - $g_3'' = \phi_3\big(Epid, Sick(X), Treat(X, m_2)\big)$
    - $g_3''' = \phi_3\big(Epid, Sick(eve), Treat(eve, m_2)\big)$



Types of Shattering

- On-demand shattering
  - Splitting on constraints only if the application of an LVE operator requires it
- E.g., given $P\big(Sick(eve)\big)$
  - Eliminate $Travel(X), Treat(X, M)$ before splitting of $Sick(eve)$
- Does not change complexity of the problem
  - May be hard to determine when the optimal point for on-demand shattering is
    - Extra work for checking



LVE: Algorithm

- Assumption:
  - Pre-emptive shattering
  - Single ground query term
    - Propositional random variable
    - Instance (grounding) of a PRV
- Inputs:
  - Model $G = \{g_i\}_{i=1}^{n}$
  - Query term Q
  - Evidence $\boldsymbol{E}$ encoded in evidence parfactors $\{g_e\}_{e=1}^{m}$
- Output:
  - Parfactor $g = \phi(Q)$
    - Encodes the a-posteriori probability distribution of Q given $\boldsymbol{E}$

LVE: Algorithm

LVE$(G, Q, \{g_e\}_{e=1}^m)$

$G \leftarrow$ Shatter G on Q, $\{g_e\}_{e=1}^m$, and on itself

$G \leftarrow$ Absorb $\{g_e\}_{e=1}^m$ in G

while G contains non-query terms **do**

if a PRV A fulfils the preconditions of $\mathrm{sum{-}out}$ **then**

$G \leftarrow$ Apply $\mathrm{sum{-}out}$ to A in G

else

$G \leftarrow$ Apply an enabling operator ($\mathrm{multiply}$, $\mathrm{count{-}convert}$, expand, $\mathrm{count{-}normalise}$, split, ground) on some parfactors in G

How to choose?

G may contain several parfactors $\phi_i(Q)$

$g \leftarrow$ Multiply all parfactors in G into one parfactor

$g \leftarrow$ Normalise the potentials in g

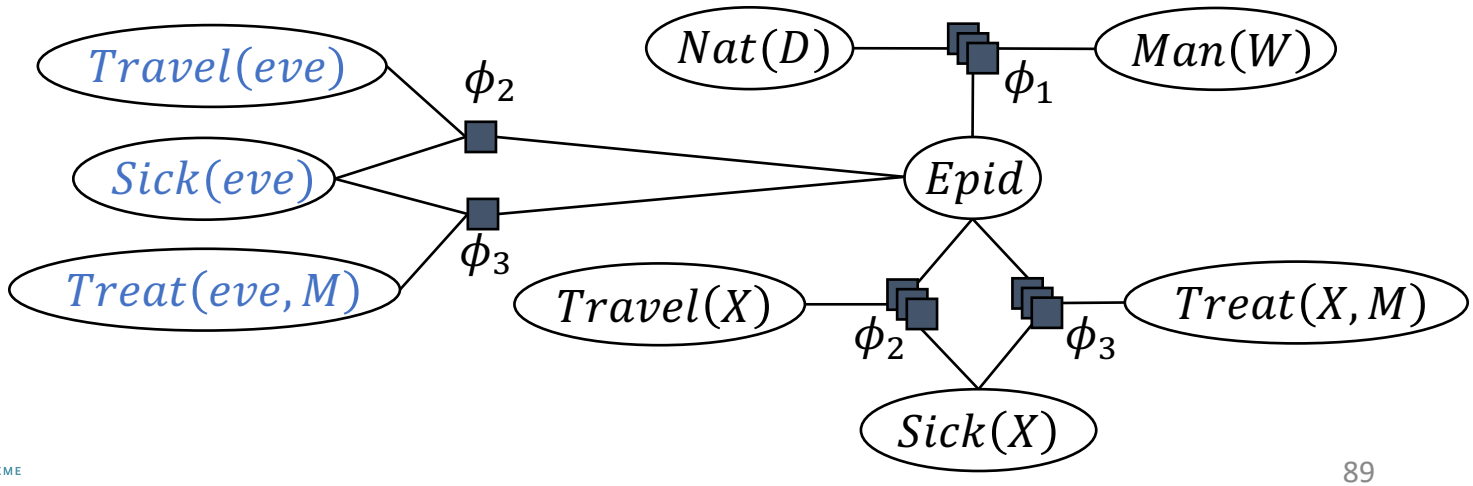
return g

LVE: Heuristics

- Important for an implementation
  - Cannot search all possible permutations of all possible operator applications
- One possible heuristics
  - Choose sum-out operations over any other operation
    - Explicitly written down in algorithm
  - Only consider multiplication if the arguments of the two parfactors are the same or ground
    - Avoid scaling
  - Choose operation that results into the smallest parfactor(s) to be added to G
    - If same size: choose at random
    - May result in sub-optimal application order or unnecessary applications
      - E.g., if a grounding is unavoidable, the heuristics may lead to various count conversions being applied before grounding as the resulting parfactor of a count conversion is usually smaller in size than the result of grounding the same logvar

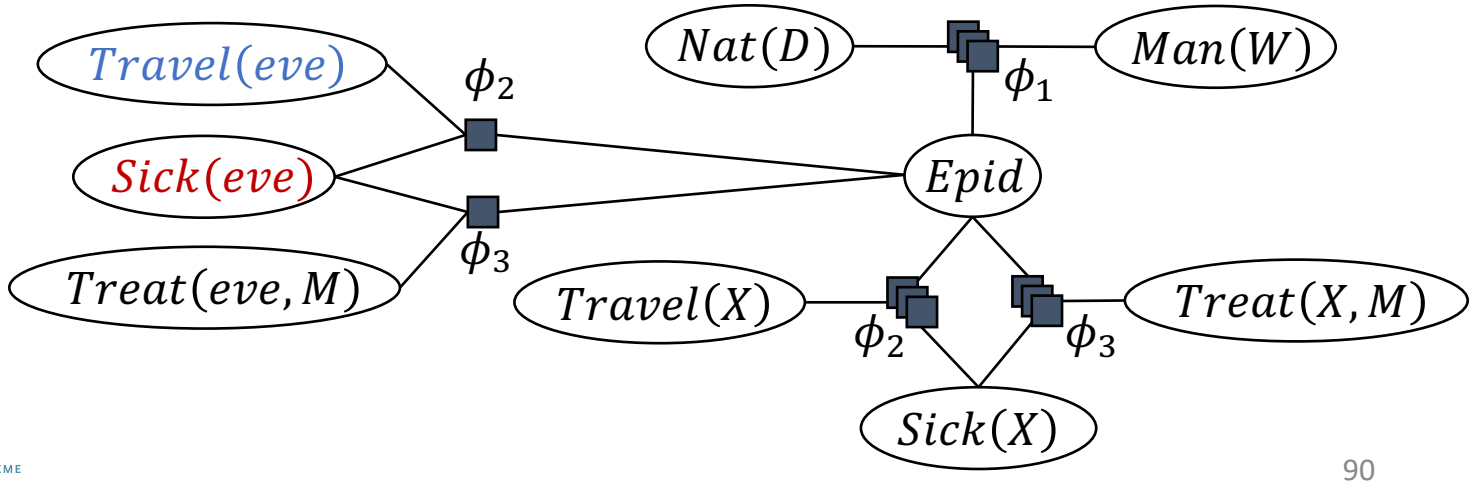
LVE: Example

- Model: $G = \{g_i\}_{i=1}^{3}$
  - T constraints
- Query term: $Travel(eve)$
- Evidence: $Sick(eve) = true$
- After shattering:



LVE: Example

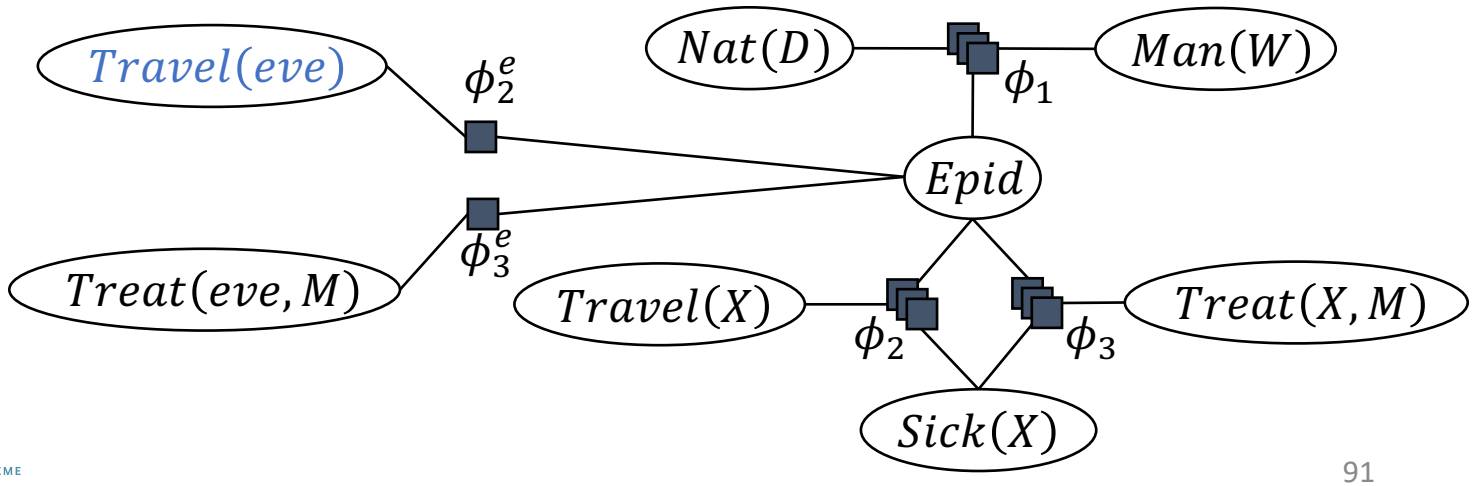
- Absorbing evidence $Sick(eve) = true$:
  - Absorb evidence in $\phi_2(Epid, Sick(eve), Travel(eve))$
    - Yields $\phi_2^e(Epid, Travel(eve))$
  - Absorb evidence in $\phi_3(Epid, Sick(eve), Treat(eve, M))$
    - Yields $\phi_3^e(Epid, Treat(eve, M))$



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Treat(eve, M)$
      - Yields $\phi_3^{e\prime}(Epid) \rightarrow$ size: $2^1 = 2$
    - $Travel(X)$
      - Yields $\phi_2'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$
    - $Treat(X, M)$
      - Yields $\phi_3'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$

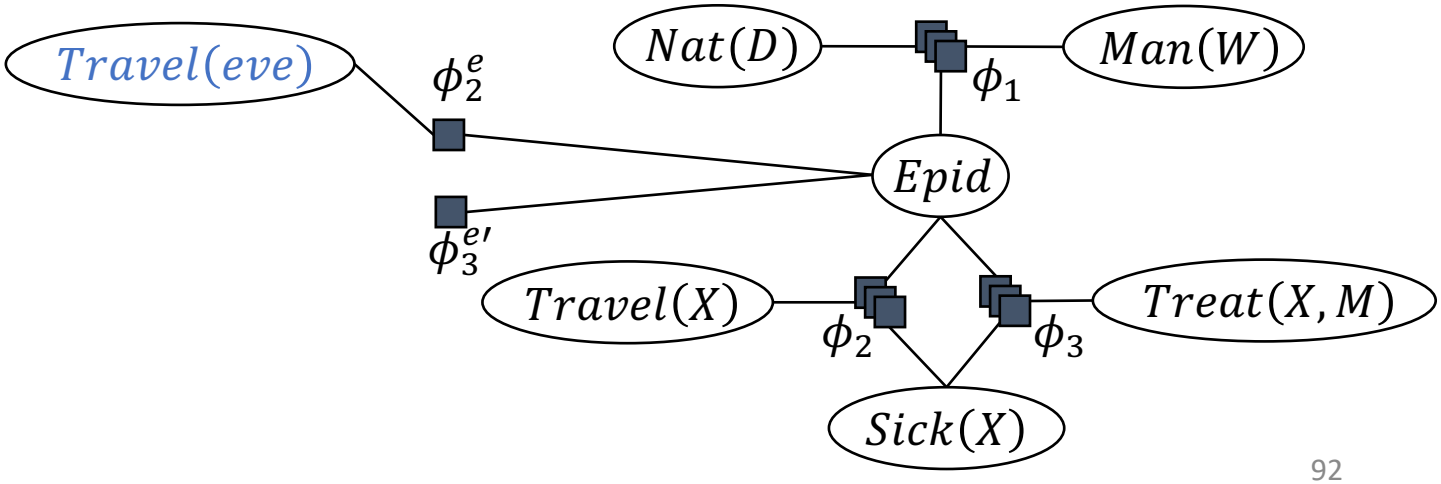
Smallest size



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Travel(X)$
      - Yields $\phi_2'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$
    - $Treat(X, M)$
      - Yields $\phi_3'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$

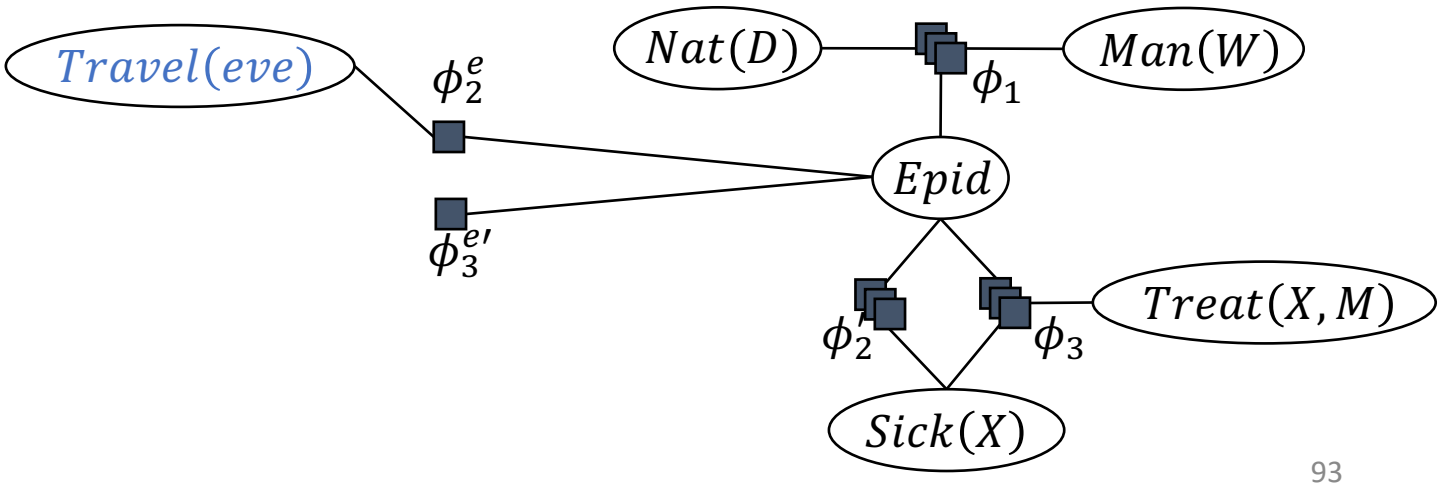
Chosen at random



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Treat(X, M)$
      - Yields $\phi_3'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$

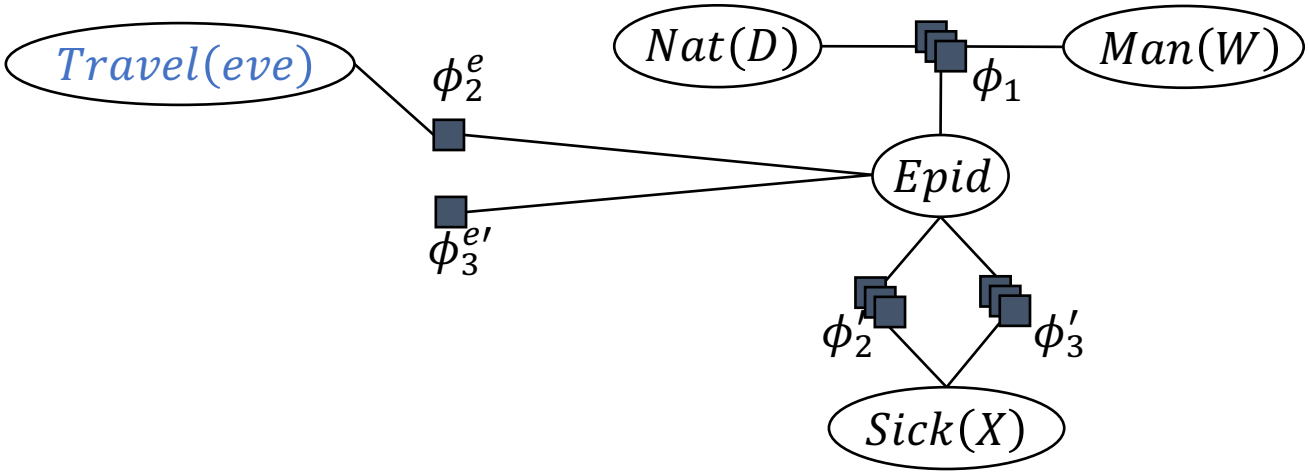
Only one



LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Multiply ϕ_2' and ϕ_3'
      - Yields $\phi_{23}'(Epid, Sick(X)) \rightarrow$ size: $2^2 = 4$
    - Multiply ϕ_2^e and $\phi_3^{e'}$
      - Yields $\phi_{23}'(Epid, Travel(eve)) \rightarrow$ size: $2^2 = 4$
    - Count-convert $Nat(D)$
      - Yields $\phi_1^D(Epid, \#_D[Nat(D)], Man(W)) \rightarrow$ size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi_1^W(Epid, Nat(D), \#_W[Man(W)]) \rightarrow$ size: $2^2 \cdot 3 = 12$

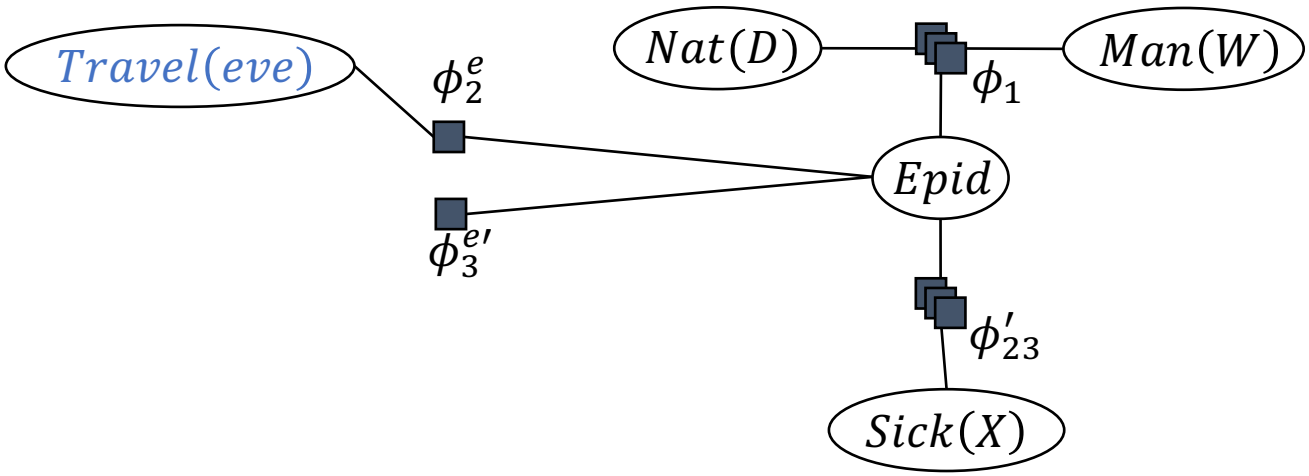
Chosen at random



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Sick(X)$
      - Yields $\phi''_{23}(Epid) \rightarrow$ size: 2

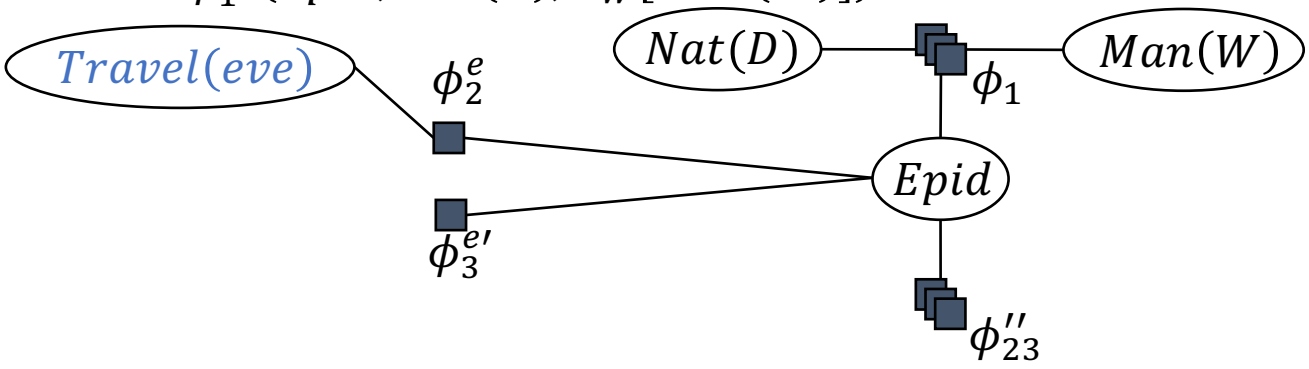
Only one



LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Multiply $\phi_3^{e\prime}$ and ϕ_{23}''
      - Yields $\phi_{23}^{e''}(Epid) \rightarrow$ size: 2
    - Multiply ϕ_2^e and $\phi_3^{e\prime}$
      - Yields $\phi_{23}'\big(Epid, Travel(eve)\big) \rightarrow$ size: $2^2 = 4$
    - Count-convert $Nat(D)$
      - Yields $\phi_1^D\big(Epid, \#_D[Nat(D)], Man(W)\big) \rightarrow$ size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi_1^W(Epid, Nat(D), \#_W[Man(W)]) \rightarrow$ size: $2^2 \cdot 3 = 12$

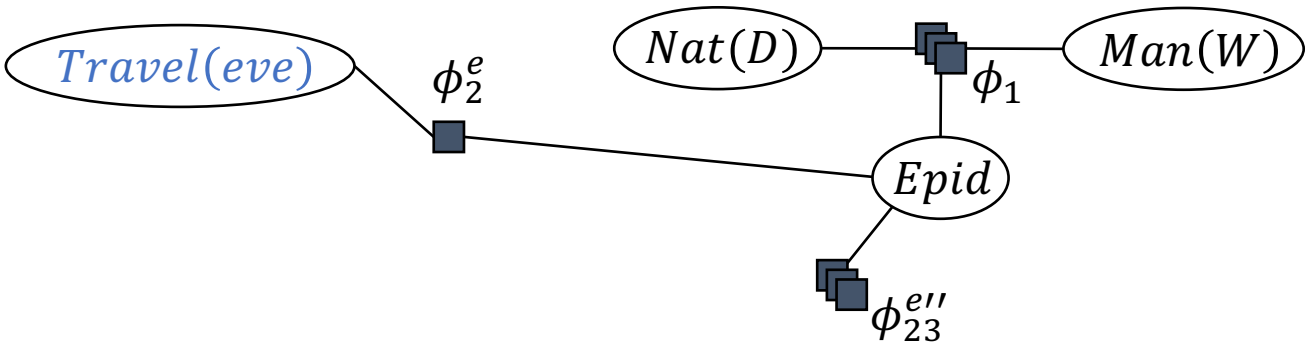
Smallest size



LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Multiply ϕ_2^e and $\phi_{23}^{e''}$
      - Yields $\phi_{23}^{e'}(Epid, Travel(eve))$ → size: $2^2 = 4$
    - Count-convert $Nat(D)$
      - Yields $\phi_1^D(Epid, \#_D[Nat(D)], Man(W))$ → size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi_1^W(Epid, Nat(D), \#_W[Man(W)])$ → size: $2^2 \cdot 3 = 12$

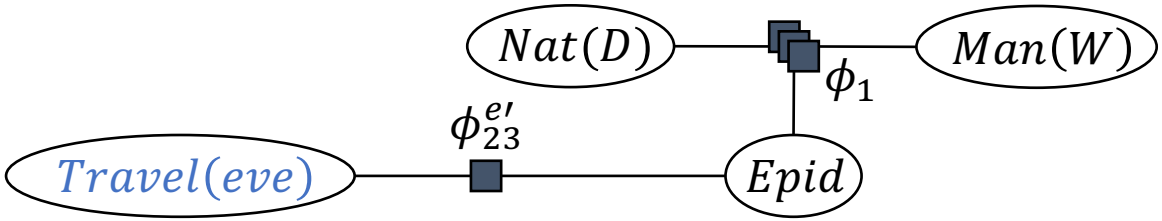
Smallest size



LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Count-convert $Nat(D)$
      - Yields $\phi_1^D\big(Epid, \#_D[Nat(D)], Man(W)\big) \rightarrow$ size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi_1^W(Epid, Nat(D), \#_W[Man(W)]) \rightarrow$ size: $2^2 \cdot 3 = 12$

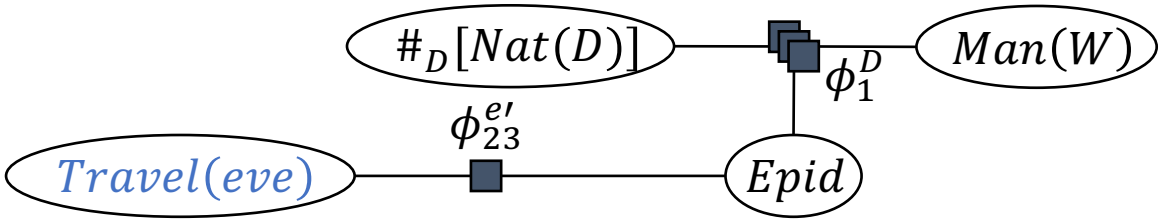
Chosen at random



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Man(W)$
      - Yields $\phi_1'(Epid, \#_D[Nat(D)]) \rightarrow$ size: $2 \cdot 3 = 6$

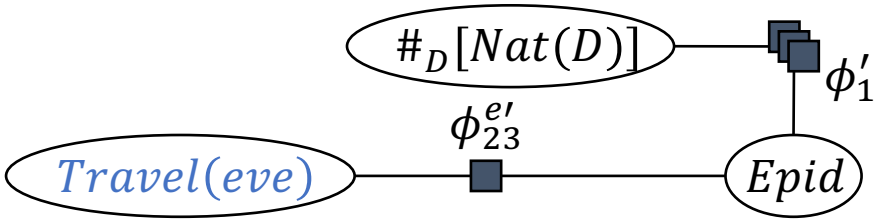
Only one



LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $\#_D[Nat(D)]$
      - Yields $\phi_1''(Epid) \rightarrow$ size: 2

Only one

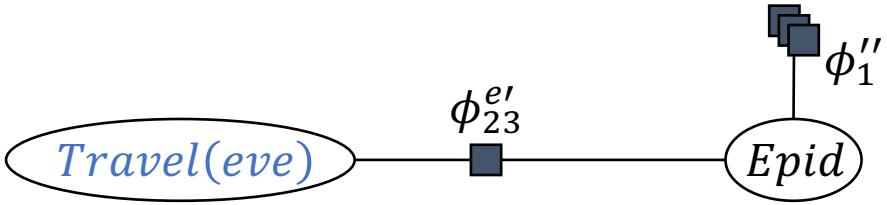


No uncounted logvars left
(basically standard VE + CRVs)

LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Multiply ϕ_1'' and $\phi_{23}^{e'}$
      - Yields $\phi_{123}^{e''}(Travel(eve), Epid)$ → size: 4

Only one

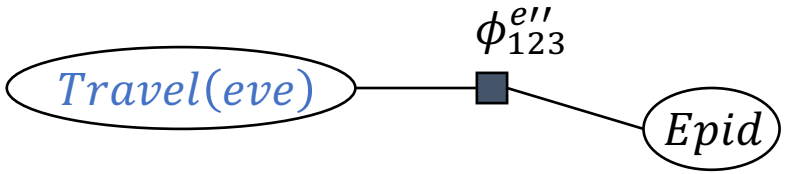


Only propositional and ground random variables left (standard VE)

LVE: Example

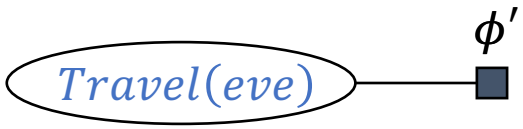
- Eliminate all non-query terms
  - PRVs fulfilling sum−out preconditions:
    - $Epid$
      - Yields $\phi'(Travel(eve))$ → size: 2

Only one



LVE: Example

- No non-query terms left
- Multiply all parfactors in G together
  - Only one parfactor $g = \phi'\big(Travel(eve)\big)$
- Normalise g
  - Yields $g = \phi\big(Travel(eve)\big)$ containing the probability distribution over $Travel(eve)$
- Return g



LVE: Example – Complete Derivation

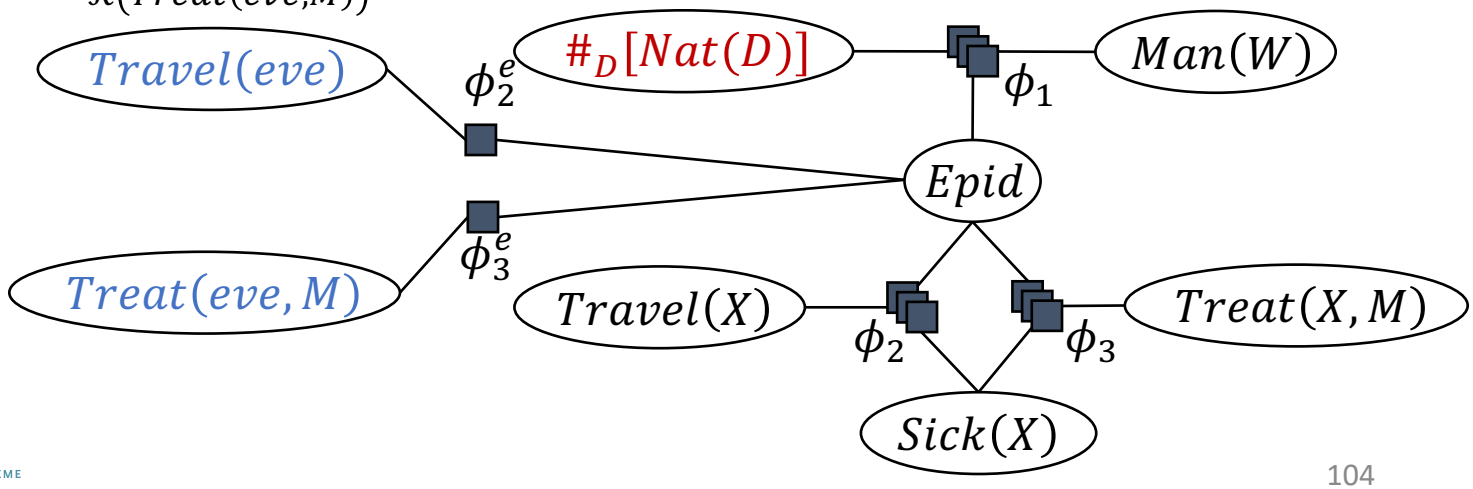
$$P\big(Travel(eve)|sick(eve)\big)$$

$$= \frac{1}{Z} \sum_{e \in \mathcal{R}(Epid)} \phi_2^e(Travel(eve), e) \sum_{h_n \in \mathcal{R}(\#_D[Nat(D)])} Mul(h_n) \left(\sum_{m \in \mathcal{R}(Man(W))} \phi_1(e, m, h_n) \right)^{|\mathcal{D}(W)|}$$

$$\left(\sum_{s \in \mathcal{R}(Sick(X))} \left(\sum_{tt \in \mathcal{R}(Treat(X,M))} \phi_3(e, s, tt) \right)^{|\mathcal{D}(M)|} \sum_{t \in \mathcal{R}(Travel(X))} \phi_2(e, s, t) \right)^{|\mathcal{D}(X)|}$$

$$\sum_{te \in \mathcal{R}(Treat(eve,M))} \phi_3(e, te)$$

- After shattering, absorption, and the required count conversion



LVE: Implementation

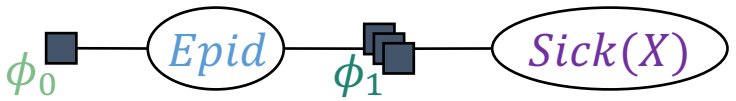
- Available at:
  - https://dtai.cs.kuleuven.be/software/gcfove
  - Includes a VE implementation for comparison
- Input: BLOG files
  - Based on Bayesian Logic Programming Language
    - https://bayesianlogic.github.io
- Differences
  - Constraint language and domains:
    - Intensional language: all domain constants apply except those explicitly excluded via ≠
    - Domains cannot be subsets of other domains
  - No explicit multiplication operator
    - Merged into sum-out operator

BLOG Input

- Components
  - Logvars
  - Domain definitions
  - Ground random variables
  - PRVs
  - Factors
  - Parfactors
- Potential lists
  - Start at all $true$
  - End at all $false$
    - If you think of the assignments as binary numbers, then the numbers are decreasing

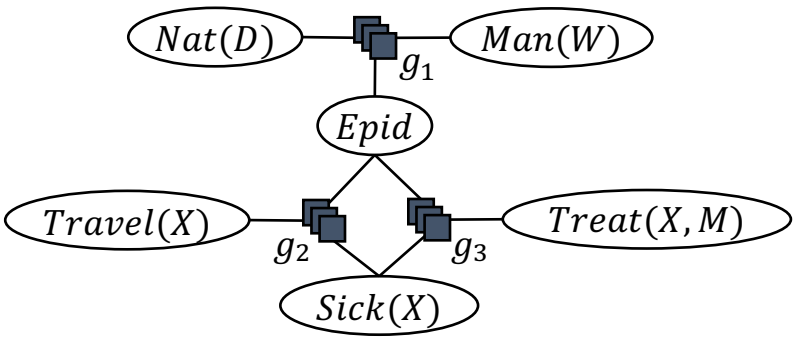
BLOG file

```
type Person;
guaranteed Person x[3];

random Boolean Epid;
random Boolean Sick(Person);

factor MultiArrayPotential[[0.1, 0.9]] Epid;

parfactor Person X. MultiArrayPotential
      [[0.5,0.6,0.7,0.8,0.9,0.7,0.5,0.3]]
      (Epid, Sick(X));

query Sick(x3); // query

obs Sick(x1)=true; // observation
```

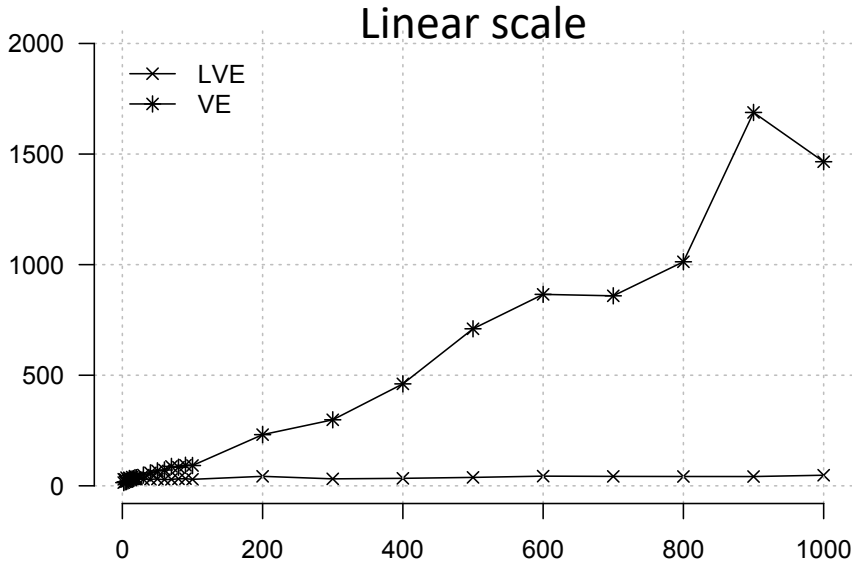


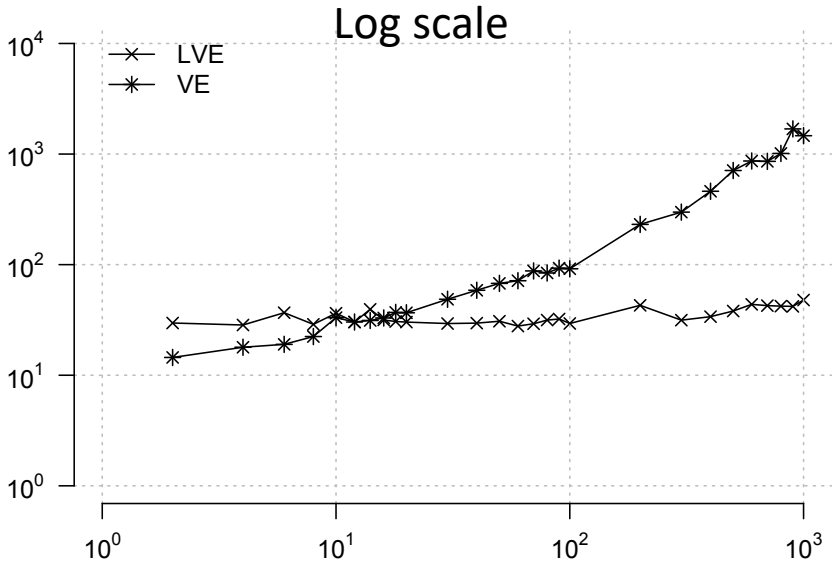
Runtimes: Increasing Domain Sizes

- Example model
  - All domain sizes 2, except $|\mathcal{D}(X)| \in \{2,4,\dots,20, 30,\dots,100,200,\dots,1000\}$
  - Query: $P\big(Travel(x_1)\big)$



Runtimes in milliseconds



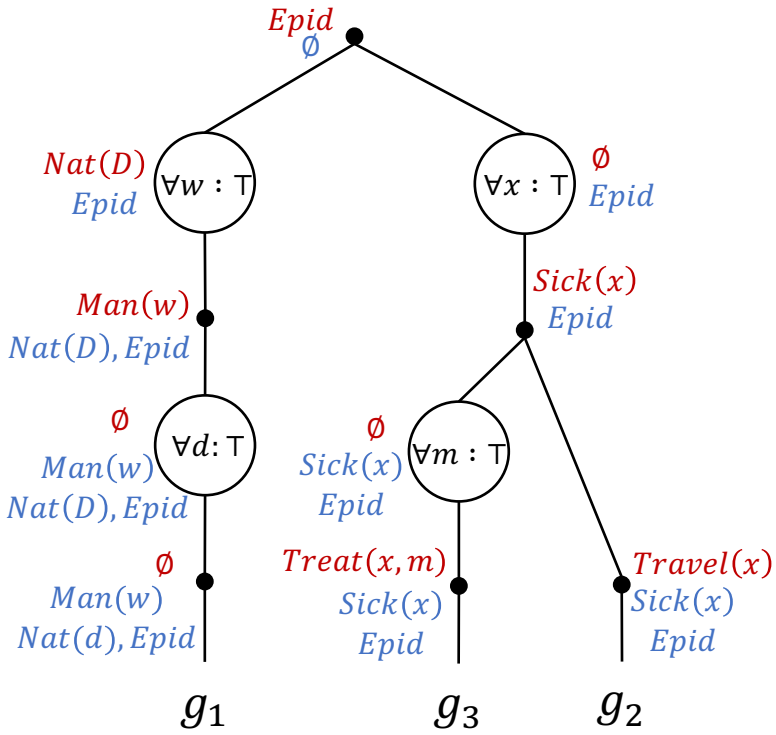


Interim Summary

- LVE Operators
  - Lifted summing out
  - Lifted multiplication
  - Count conversion
  - Lifted absorption
  - Splitting
    - Other based on splitting: expand, ground, count-normalise
- Shattering = splitting on
  - Query terms, evidence, model constraints
- LVE algorithm
  - Heuristics
  - Implementation

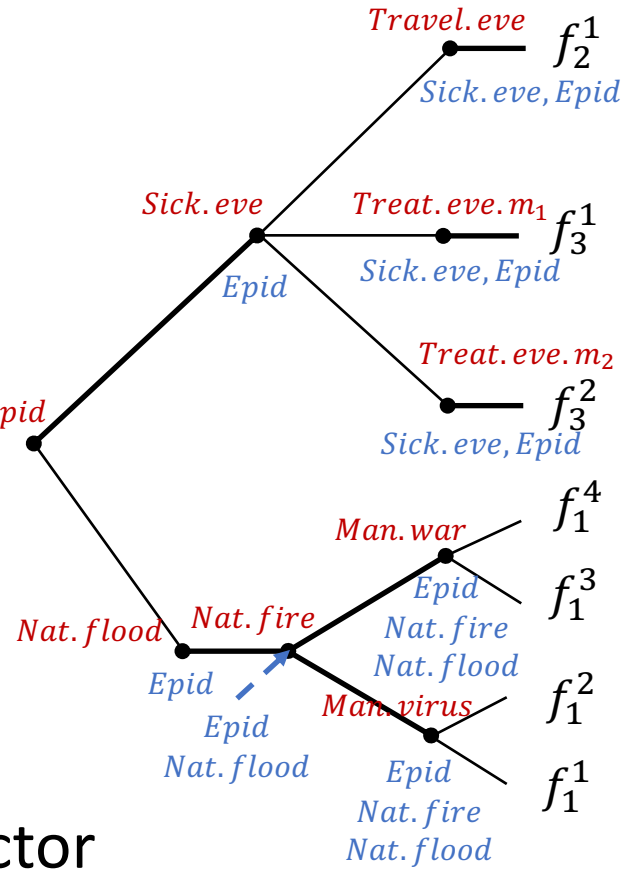
Theoretical Analysis

Lifted Variable Elimination



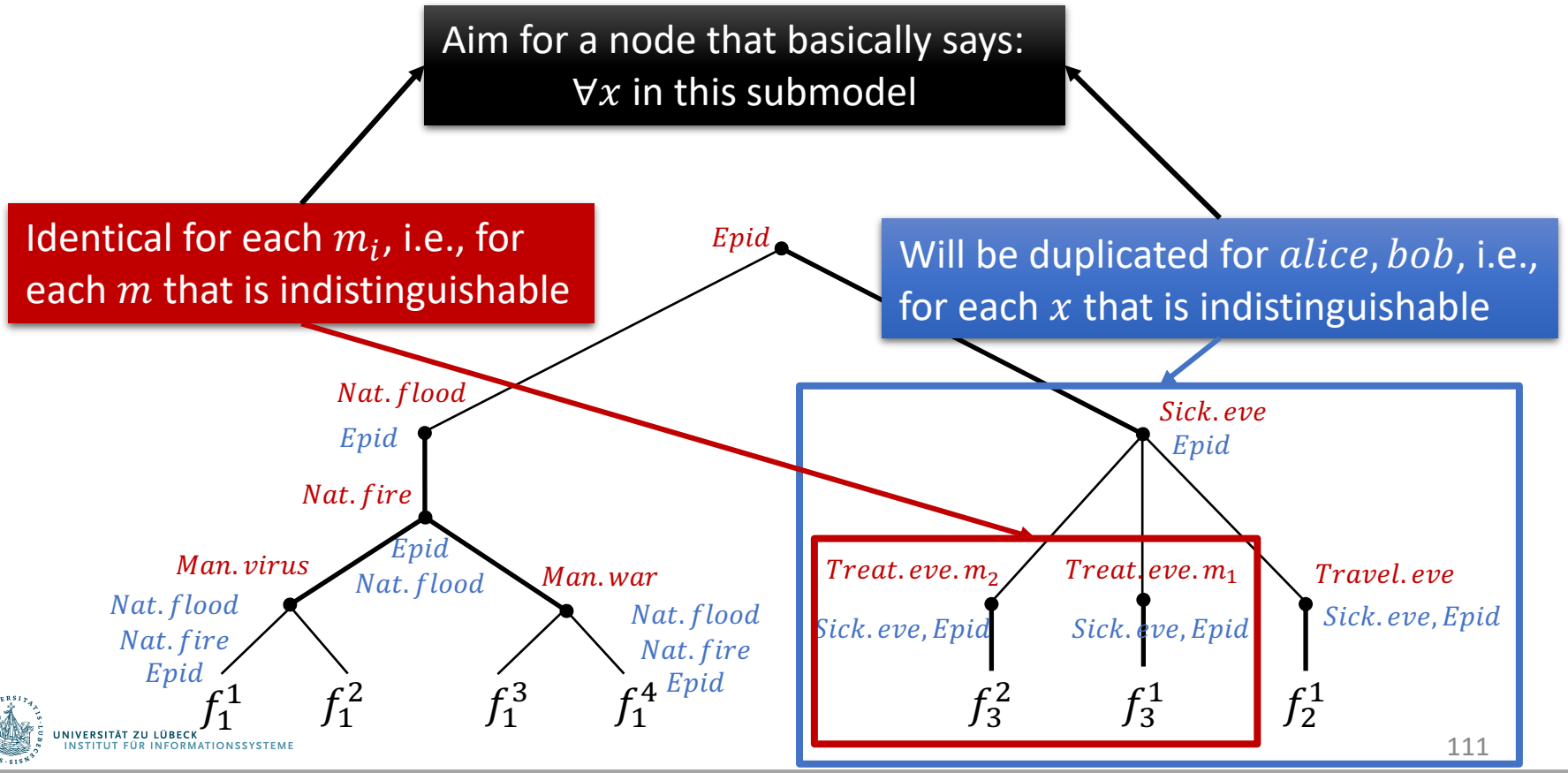
Complexity

- Remember:
  - Decomposition tree (dtree)
    - To represent VE calculation as a tree
    - Properties: cutset, context, cluster
      - Size of largest cluster = tree width
  - Runtime complexity of VE:
    $$O(n_T \cdot r^w)$$
    - Number of eliminations/inner nodes $n_T = |\boldsymbol{R}|$
    - Maximal range $r = \max_{R \in \boldsymbol{R}} |\mathcal{R}(R)|$
    - Tree width w
- *Informally*, LVE complexity still worst case size of an (intermediate) factor *times* # of eliminations
  - Use a lifted dtree, called first-order dtree (FO dree), to get worst case size of an (intermediate) factor using lifted width



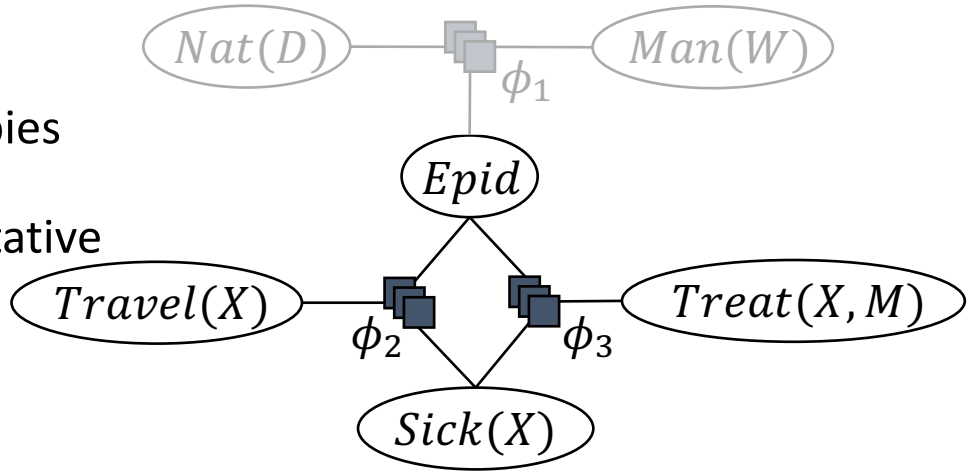
Dtree with Repeating Structures

- Decomposition of $gr(G)$
  - Duplicated subtrees where lifted summing out applies → in FO dtree: represent only once!



Decomposition into Partial Groundings

- Introduce a new inner node: DPG node denoted $\forall\, x : C$
  - DPG = Decomposition into partial groundings
    - Replaces a logvar with a representative constant
    - The resulting model/subtree is identical for each constant represented
      - Allows for considering the resulting model without the grounded logvar for further decomposition (top-down)
  - E.g., submodel below $Epid$ in the graph
    - Logvar X appears in each parfactor
    - Grounding X leads to copies of the same submodel
    - Replace X with representative $x \rightarrow$ partial grounding
      - Whatever you do to x applies to all constants represented



  - Represent that $\forall\, x : C$, C a constraint, the subtree below would be identical

DPG Definition

- Assume that (sub)model G fulfils a *normal form** where
  - Domains are either disjoint or identical
  - Logvars share the same name if they refer to the same domain over different parfactors
  - Constraints are $\top$
    - Formal definition by Taghipour includes inequality constraints

  $\Rightarrow$ No further splitting operations necessary (split, expand, count-normalise)
- Decomposition into partial groundings of model G by logvar X with $\forall g \in G : X \in lv(g)$

$$DPG(G, X) = \bigcup_{g \in G} g\theta \,, \theta = \{X \to x\}$$

* possible to rewrite any model in polynomial time into normal form

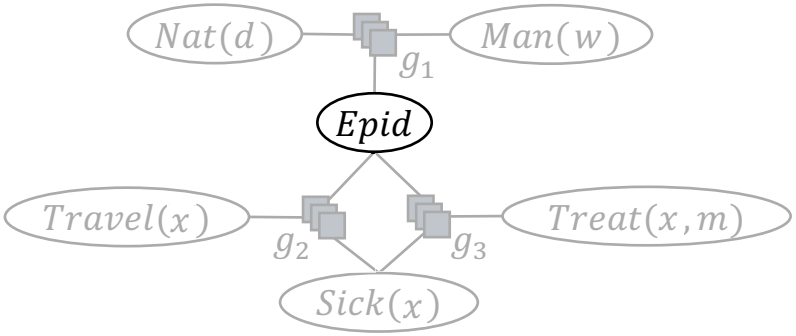
FO Dtree Construction

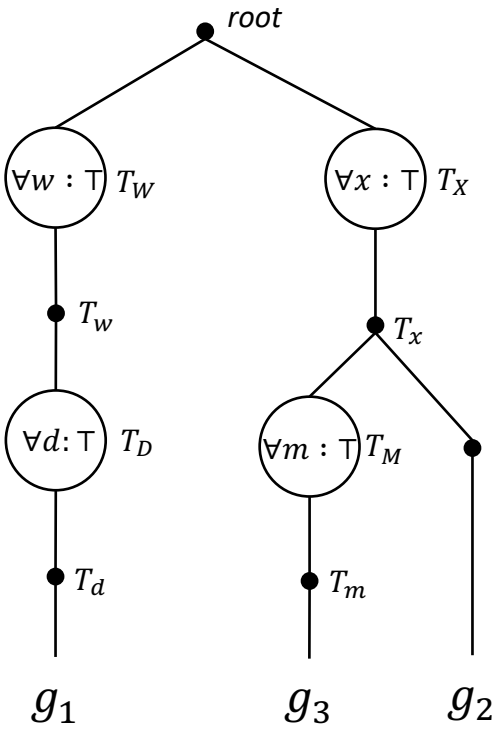
- Recursively, starting with G as the current model G' at the root
  - Check if there exists logvar X that allows for a DPG in G'
    - If so, make current node a DPG node T_X for X, replace X with representative x, i.e., apply $\theta = \{X \to x\}$ to G', add child node T_x with $G' = G\theta$ as current model
      - If parent node is a DPG node $T_{X'}$ as well, with current node being $T_{x'}$, add new DPG node T_X as child of $T_{x'}$
  - Otherwise: Partition G' w.r.t. to logvars (if exist) or random variables into $\{G'_i\}_{i=1}^n$, with $G' = \bigcup_{i=1}^n G'_i$
    - Add a child node for each G'_i with G'_i as current model
- Until
  - All logvars replaced by representatives

  and
  - Only one parfactor per partition

FO dtree: Example

- Root: In G, no logvar for a DPG
  - Partition based on, e.g., X
    - $G_0 = \{g_2, g_3\}, G_1 = \{g_1\}$
  - Right: $G_0 = \{g_2, g_3\}$
    - DPG with X
    - Replace X with x
    - No logvar for DPG
    - Partition based on M
      - $G_{01} = \{g_2\}, G_{02} = \{g_3\}$
    - Right: $G_{01} = \{g_2\}$
      - No logvars and only one parfactor left
    - Left: $G_{02} = \{g_3\}$
      - DPG with M
        - Replace M with m
        - No logvars and only one parfactor left
  - Left: $G_1 = \{g_1\}$
    - DPG with W
    - Replace W with w
    - DPG with D
      - Replace D with d
      - No logvars and only one parfactor left





FO Dtree Definition

- An FO dtree has three node types
  - DPG node T_X
    - Represents a DPG (top-down)
    - Given by a tuple (X, x, C) with X a logvar, x a representative constant, and C a constraint
      - In this lecture: $C = \top$
      - Denoted $(\forall x : C)$ in graphical representation of the tree
  - VE node T
    - Represents a partitioning
    - All inner nodes that are not DPG nodes
  - Leaf node L
    - Contains a parfactor, grounded with representative constants

FO Dtree Definition

- Let $DPG, VE, Leaf$ be the sets of all DPG, VE, leaf nodes, respectively
- Then, an FO dtree T for a model G is given by $T = (V, E)$ where
  - $V = DPG \cup VE \cup Leaf$
  - $E = (DPG \times VE) \cup (VE \times DPG) \cup (VE \times VE) \cup (VE \times Leaf)$
    - A VE node can follow a DPG and a VE node, a DPG and a leaf node can follow a VE node
  - Further
    - Each DPG node T_X has a child VE node T_x whose model is G_x is a representative model of G_X with $G_x = G_X\theta, \theta = \{X \to x\}$
    - Each leaf with representative constant x has in its parfactor descends from exactly one DPG node $T_X = (X, x, C)$
    - Each leaf descending from DPG node $T_X = (X, x, C)$ has representative constant x in its parfactor
      - Effect:
        At beginning of construction, one would have to partition initial model G into one partition of parfactors containing only random variables and one partition of parfactors containing logvars

FO dtree Properties

- Property definitions as before
  - Cutset

$$cutset(T) = \left(\bigcup_{T_i, T_j \in children(T)} rv(T_i) \cap rv(T_j) \right) \setminus acutset(T)$$

$$acutset(T) = \bigcup_{T' \in ancestor(T)} cutset(T')$$

  - Context

$$context(T) = rv(T) \cap acutset(T)$$

  - Cluster

$$cluster(T) = cutset(T) \cup context(T)$$

- Definition for a DPG node T_X with logvar $X_{|\mathsf{T}}$
  - $children(T_X) = \{T_{x\theta} | T_x \text{ is child of } T_X \wedge \theta \in \Theta_X\}$
    - Θ_X all grounding substitutions of $X_{|\mathsf{T}}$
  - $children(T_x) = \{T_{Y\theta} | T_Y \text{ is child of } T_x \wedge \theta \in \Theta_Y\}$
    - Child of T_x is a DPG node again, T_Y

FO dtree Properties: Example

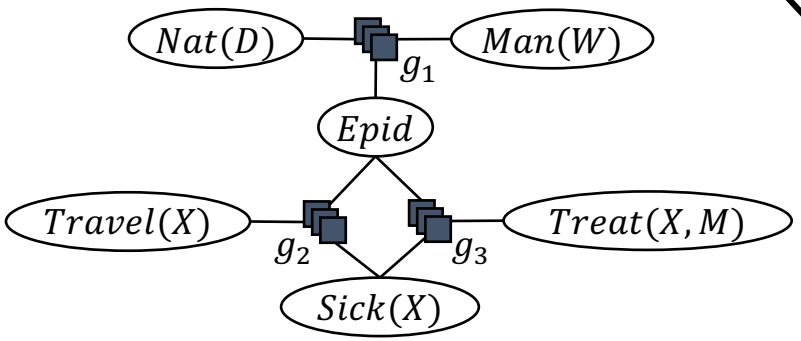
- Cutset

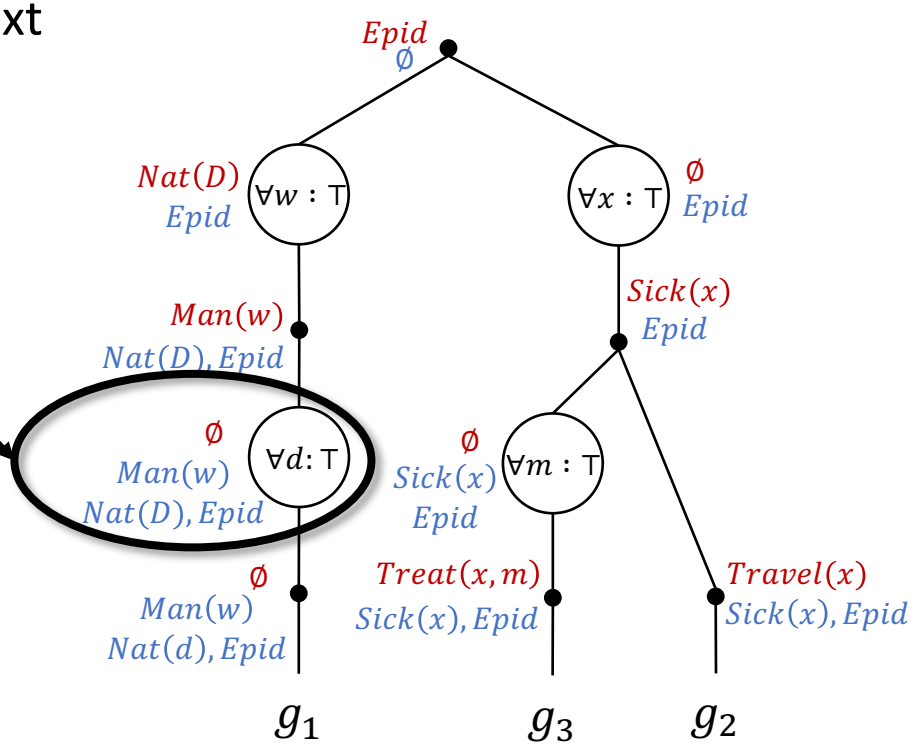
$$cutset(T) = \left(\bigcup_{T_i, T_j \in children(T)} rv(T_i) \cap rv(T_j) \right) \setminus acutset(T)$$

- Context

$$context(T) = rv(T) \cap acutset(T)$$

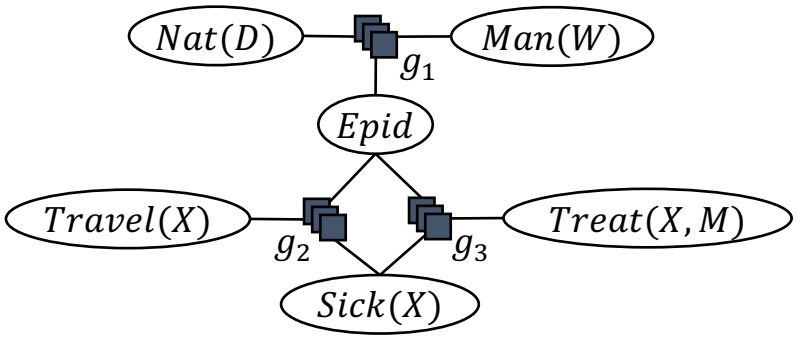
- If DPG logvar occurs in the context of its DPG node:
  - Count conversion necessary!

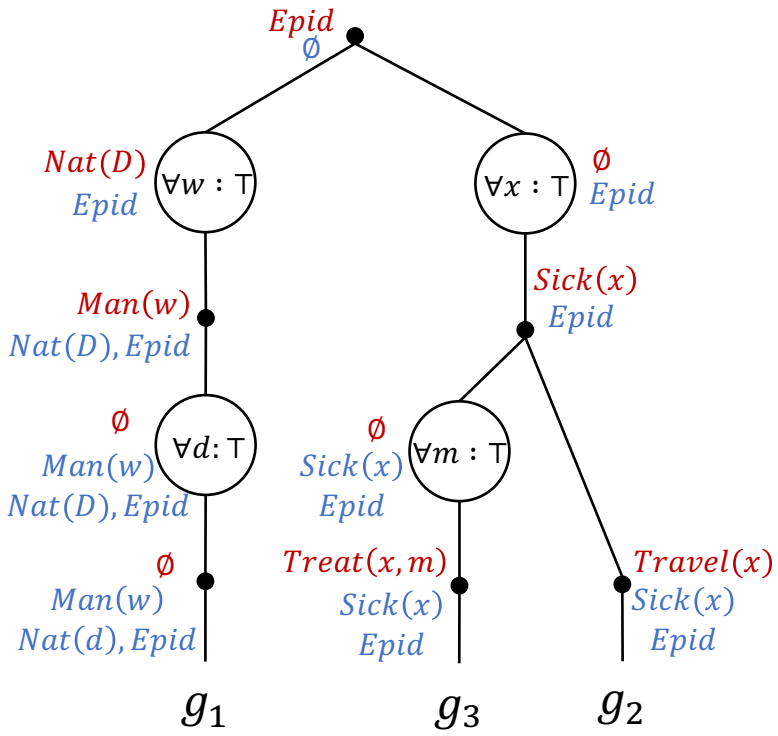




FO dtree: Bottom-up Interpretation

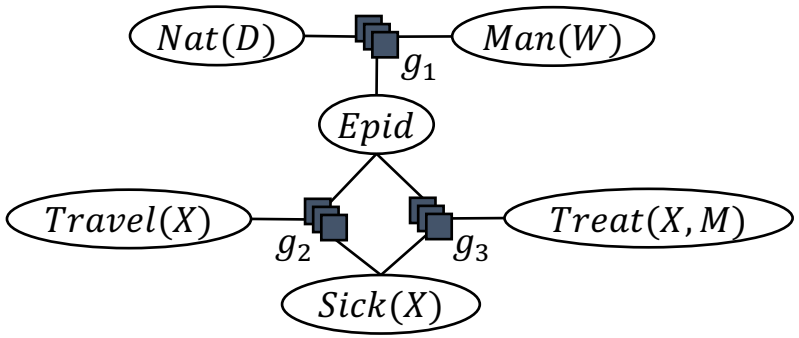
- If only lifted summing out and multiplication involved
  - VE node
    - (Multiplication), elimination
  - DPG node
    - Exponentiation
  - Cutset and context interpretation as before
    - Cutset of VE node: PRV summed out in representative summing out
    - Cutset of parent DPG node: Empty as no PRV eliminated but result exponentiated for logvar of DPG node
    - Context: all other PRVs involved at this point in operation



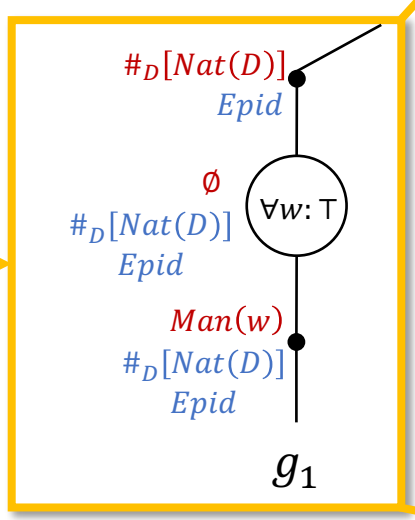


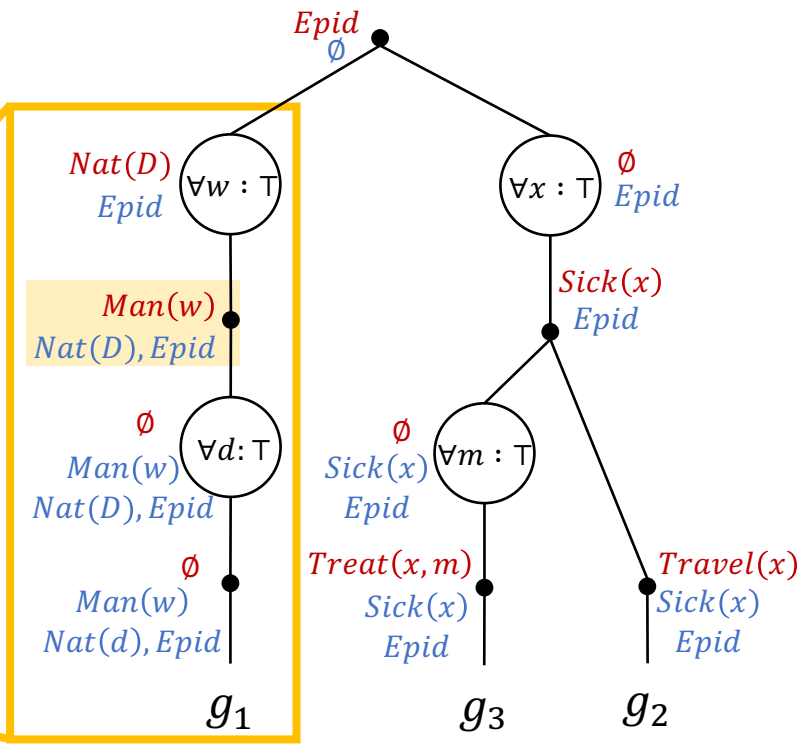
FO dtree: Bottom-up Interpretation

- No direct interpretation as LVE operations if count conversion involved
  - Shows “only” that PRV not directly eliminable
    - Counting (or grounding) necessary
    - E.g., $Nat(D)$ in context of VE node for eliminating $Man(w)$



Rework FO dtree to represent calculation directly

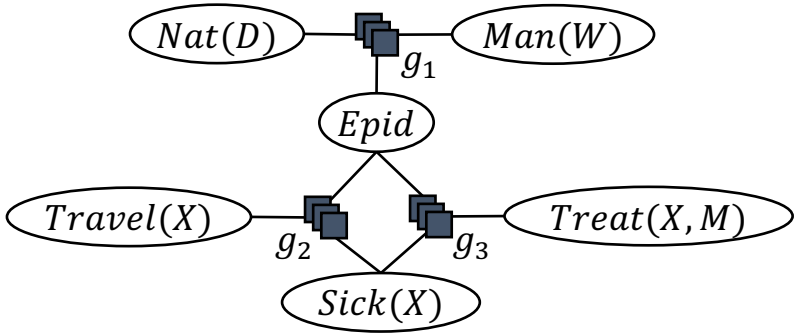


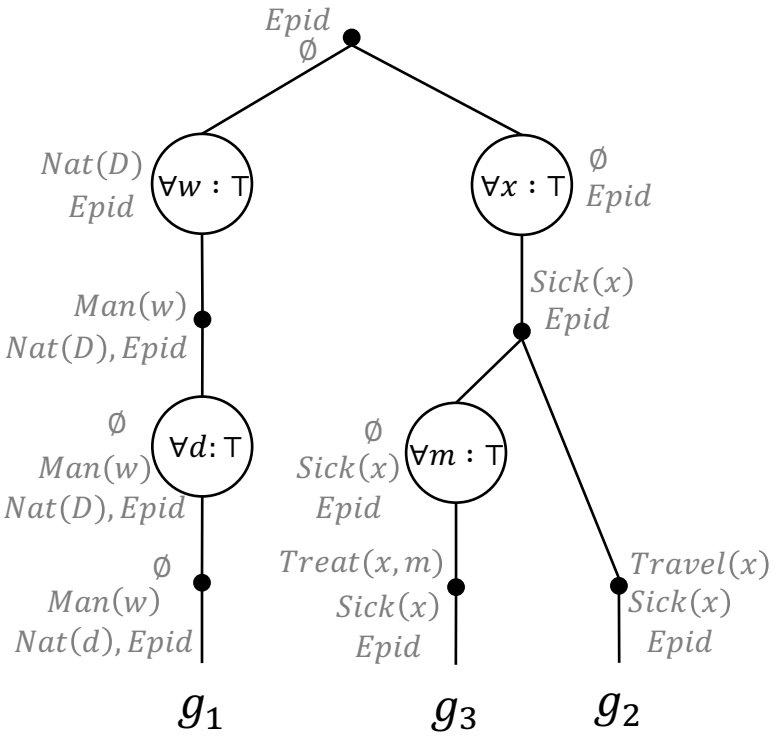


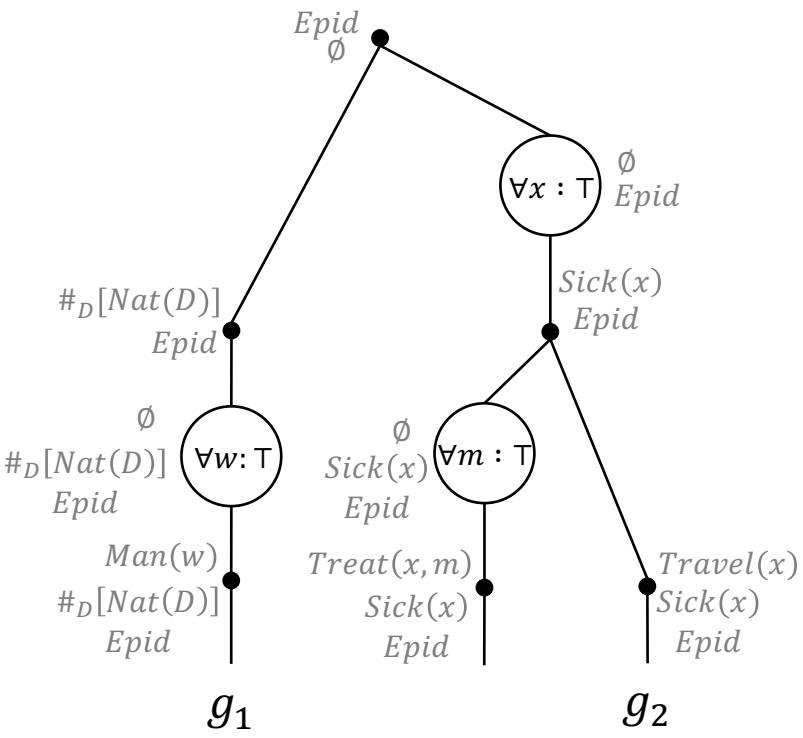
Liftability

- Given an FO dtree T for a model G
  - If the clusters of T only consist of *PRVs with representative constants* and *PRVs with one logvar*, then G has a lifted solution
    - Lifted solution: no groundings necessary; only lifted calculations ($\mathrm{sum{-}out}$, $\mathrm{multiply}$, $\mathrm{count{-}convert}$)
    - PRVs only with representative constants → lifted summing out possible
    - PRVs with one logvar → count conversion necessary (and possible)
      - Called countable
  - T is called liftable
  - Apply the count conversions to the countable logvars, transforming G → resulting FO dtree T' called counted
- For complexity analysis: Concentrate on models with liftable (counted) FO dtrees
  - Otherwise: the worst case is grounding G and performing VE

- In the example FO dtree
  - Only PRVs with representative constants or one logvar in the clusters
    → liftable
  - If applying the count conversion to D and reworking the FO dtree
    → counted, liftable

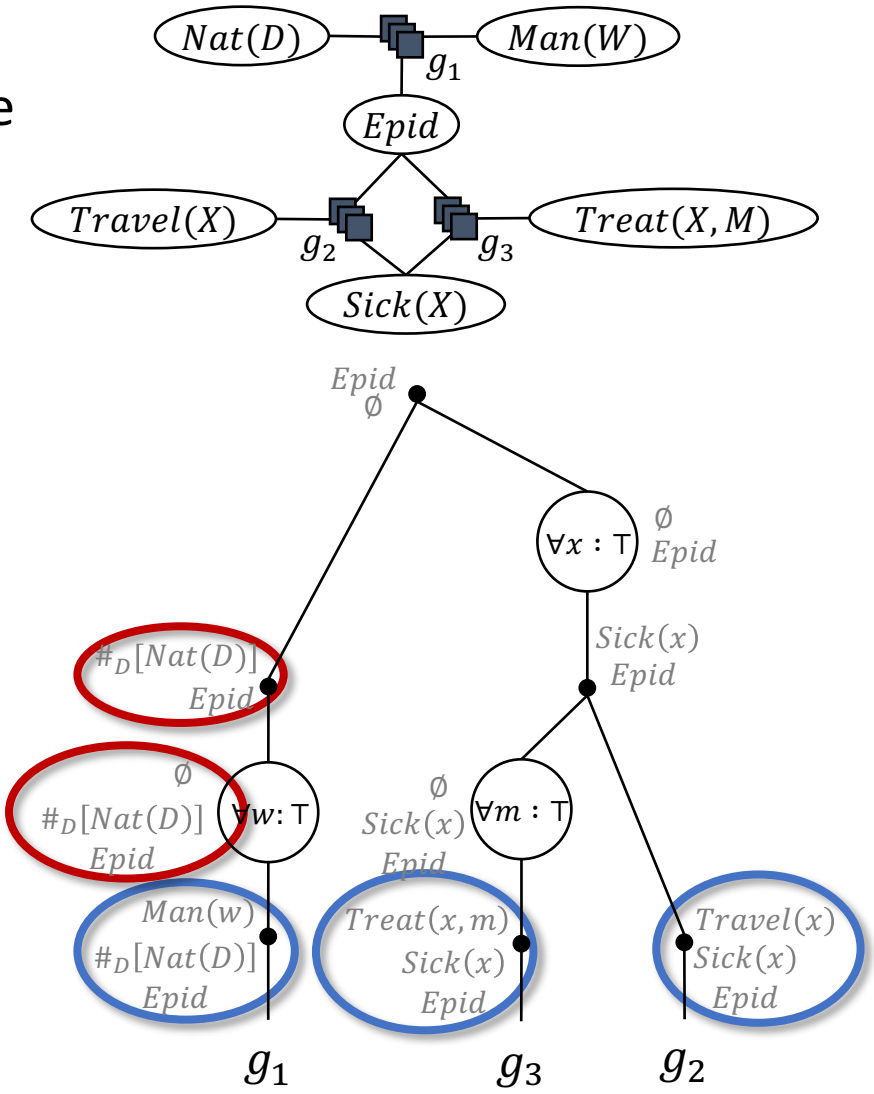






Lifted Width

- Lifted notion of tree width
  - Bound worst-case parfactor size
- Given liftable, counted FO dtree T
  - Clusters for each node in T
- Lifted width $w_T = (w_g, w_\#)$
  - w_g largest ground width
    - Largest number of PRVs with representative constants in any cluster
      - E.g., $w_g = 3$
  - $w_\#$ largest counting width
    - Largest number of CRVs in any cluster
      - E.g., $w_\# = 1$



Worst-case Parfactor Size

- Given lifted width $w_T = (w_g, w_\#)$ of a liftable, counted FO dtree T
  - E.g., $w_T = (w_g = 3, w_\# = 1)$
- Worst-case parfactor size:
  - Worst case: $w_g + w_\#$ variables in one parfactor
  - Worst-case range size for the w_g PRVs
$$r = \max_{A \in rv(G)} |\mathcal{R}(A)|$$
  - Worst-case range size for the $w_\#$ CRVs
$$\binom{n_\# + r_\# - 1}{n_\# - 1} \leq {n_\#}^{r_\#}$$
    - $n_\#$ largest domain size of any of the counted/countable logvars
    - $r_\#$ largest range size of any of the PRVs in the CRVs/one-logvar PRVs
  - Number of mappings by w_g and $w_\#$:
$$r^{w_g} \cdot ({n_\#}^{r_\#})^{w_\#} = r^{w_g} \cdot {n_\#}^{r_\# w_\#}$$
    - E.g., $2^3 \cdot |\mathcal{D}(D)|^{2 \cdot 1} = 2^3 \cdot 2^2 = 8 \cdot 4 = 32 > 12$ (actual largest size)

Complexity

- Worst-case parfactor size $r^{w_g} \cdot {n_\#}^{r_\# w_\#}$
  - E.g., $2^3 \cdot |\mathcal{D}(D)|^{2 \cdot 1} = 2^3 \cdot 2^2$
- Complexity of lifted operations
  - Lifted elimination:
    - Multiplication (goes through each line of each parfactor): $O(r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
    - Summation (goes through each line): $O(r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
    - Exponentiation (goes through each line): $O(\log_2(n) \cdot r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
      - n largest overall domain size
  - Count conversion (goes through each line of parfactor):
    - Multiplication and exponentiation: $O(\log_2(n_\#) \cdot r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
- Complexity of LVE given a liftable FO dtree T

$$O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$$

  - n_T: number of inner nodes in T
  - w_g: bounded from below by $\max_{\phi(A_1,\ldots,A_k) \in G} k$

Query Term and Evidence

- Query term: assumed one query term per query
  - Query term does not have an impact on algorithm
    - Split of at beginning
    - Introduces only a handful of propositional random variables whose elimination operations are unimportant compared to the rest of the model
- Evidence
  - If handling evidence at beginning (absorption), then the resulting model can be analysed as given
  - If considering the model without evidence, then evidence can lead to $|\mathcal{R}(A)|$ different groups per PRV
    - Multiplies the number of PRVs in a model
    - Does not change the lifted width of a model
  - CAUTION: Evidence on PRVs with arbitrarily many logvars can break symmetries and lead to groundings

Comparison

- Complexity of LVE given a liftable, counted FO dtree T for a counted model G:

$$O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$$

  - $n_T = |rv(G)| + |lv(G)|$
- Complexity of VE:

$$O\left(n_{gr(T)} \cdot r^{w}\right)$$

  - $n_{gr(T)} = \left|gr\left(rv(G)\right)\right|$
- If no count conversions involved, i.e., $w_\# = 0$,
  - $n^{r_\# \cdot 0} = 1 \rightarrow O(n_T \cdot \log_2(n) \cdot r^{w_g})$
  - $w = w_g$
  - Difference in $\log_2(n)$ for lifted computations and $n_{gr(T)}, n_T$
    - The more noticeable, the more domain sizes increase ($n_{gr(T)} \gg n_T$)

Comparison

In the lifted case, domain size n no longer occurs in an exponent whereas it does in the propositional case thanks to count conversion

- If count conversions involved, i.e., $w_{\#} > 0$,
  - $w \gg (w_g + w_{\#})$
    - CRV with a counted logvar of domain size n appears grounded in a factor
  - With one count conversion,

$$O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_{\#}}) \text{ vs. } O\left(n_{gr(T)} \cdot r^{n+c}\right)$$

    - c the number of random variables also occurring in the cluster
    - E.g., with $E = Epid$; $c = 2$:

$$\phi_1(E, Nat(D), Man(W)) \xrightarrow{\#} \phi_1^{\#}(E, \#_D[Nat(D)], Man(W)) \xrightarrow{\Sigma} \phi_1(E, \#_D[Nat(D)])$$

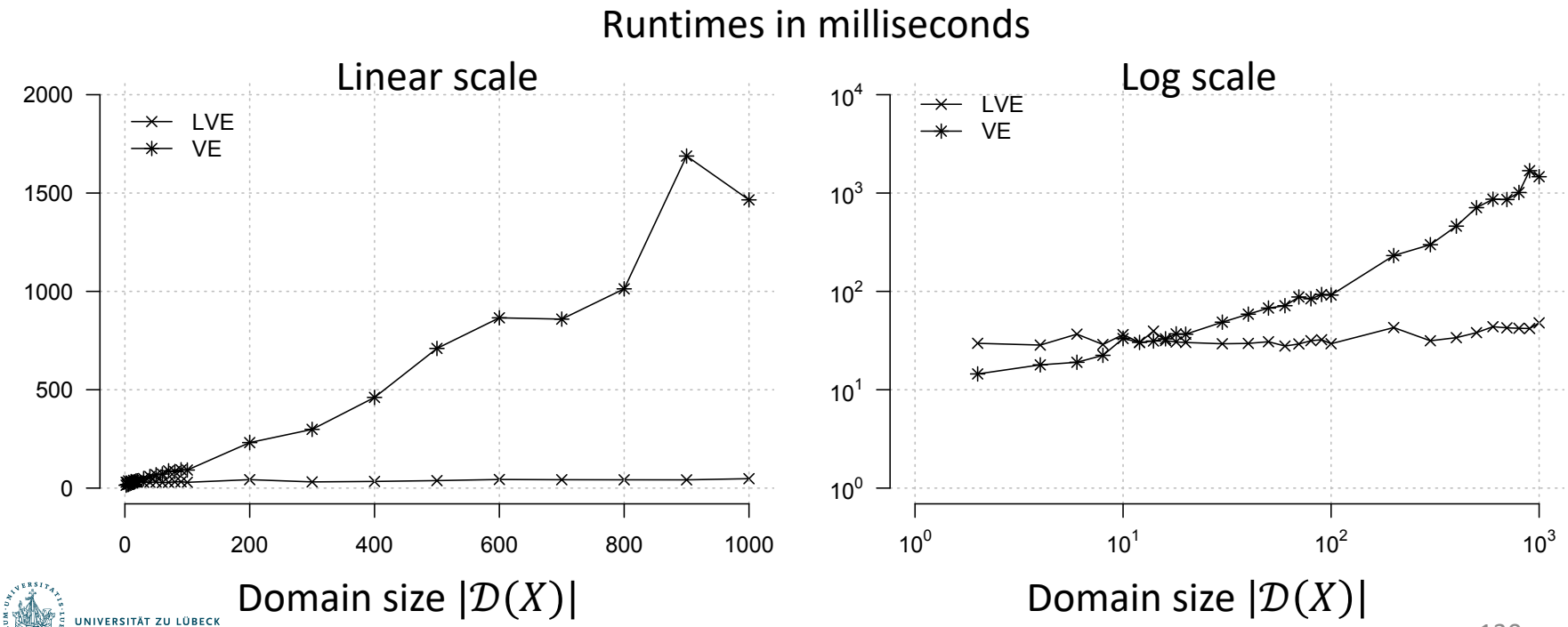
$$\left.\begin{matrix} \phi_1(E, Nat(d_1), Man(w_1)) \\ \vdots \\ \phi_1(E, Nat(d_n), Man(w_1)) \end{matrix}\right\} \xrightarrow{\cdot} \phi_1^1(E, Nat(d_1), \dots Nat(d_n), Man(w_1)) \xrightarrow{\Sigma} \phi_1^1(E, Nat(d_1), \dots Nat(d_n))$$

$$\vdots$$

$$\left.\begin{matrix} \phi_1(E, Nat(d_1), Man(w_m)) \\ \vdots \\ \phi_1(E, Nat(d_n), Man(w_m)) \end{matrix}\right\} \xrightarrow{\cdot} \phi_1^1(E, Nat(d_1), \dots Nat(d_n), Man(w_m)) \xrightarrow{\Sigma} \phi_1^1(E, Nat(d_1), \dots Nat(d_n))$$

Comparison: Runtime

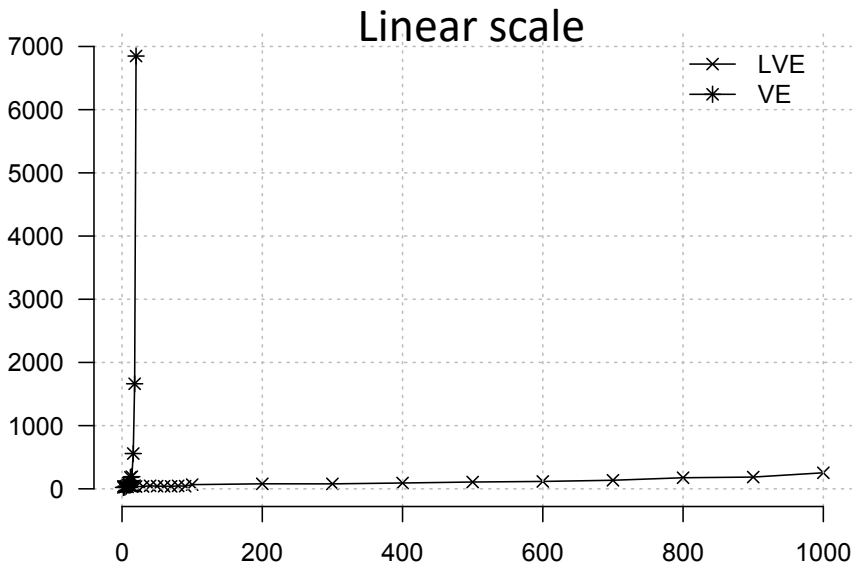
- One count conversion, i.e., $w_{\#} = 1$,
  - $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_{\#}})$ vs. $O\left(n_{gr(T)} \cdot r^{n+c}\right)$
  - Consider domain size of counted logvar constant:

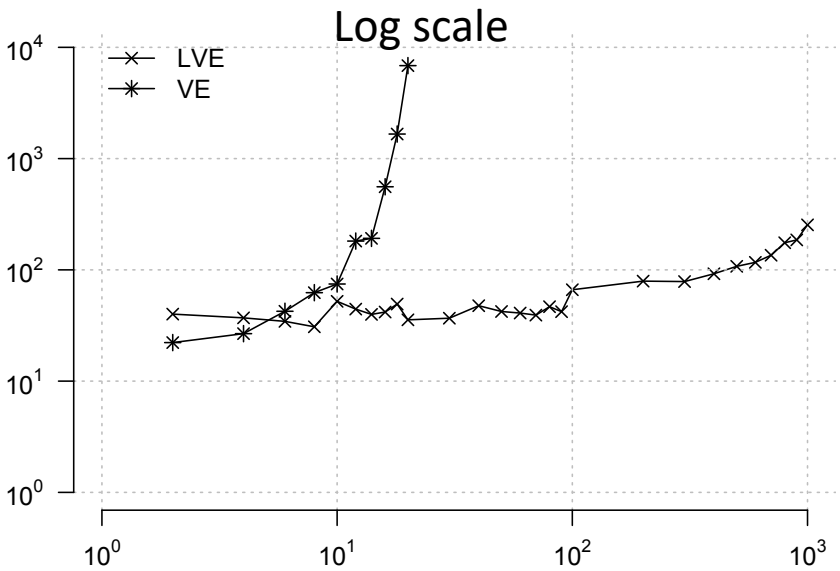


Comparison: Runtime

- One count conversion, i.e., $w_{\#} = 1$,
  - $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_{\#}})$ vs. $O\left(n_{gr(T)} \cdot r^{n+c}\right)$
  - With domain size of D and W (in g_1) increasing

Runtimes in milliseconds





Tractability

- A query answering problem is tractable
  - if it is solved by an efficient algorithm, running in time polynomial in the number of random variables
- Assume that the number of random variables is characterised by domain size n and
  - In LVE, n does not occur in the exponent
  - Solving a query answering problem is tractable under liftability
  - Runtime still exponential in other terms $(w_g, w_\#, r_\#)$
- More general results by

  Mathias Niepert and Guy Van den Broeck. Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In *AAAI-14 Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.
  - Tractability through Exchangeability

Completeness

- Class of models $\mathcal{M}$
  - Set of all possible models given some *model characteristic*
- An algorithm is complete for a class of models $\mathcal{M}$ iff
  - No groundings necessary in all models of $\mathcal{M}$
    - All models allow for a liftable FO dtree
  - Then, class called liftable
- Existing liftable classes
  - $\mathcal{M}^{2lv}$: Maximum of two logvars per parfactor

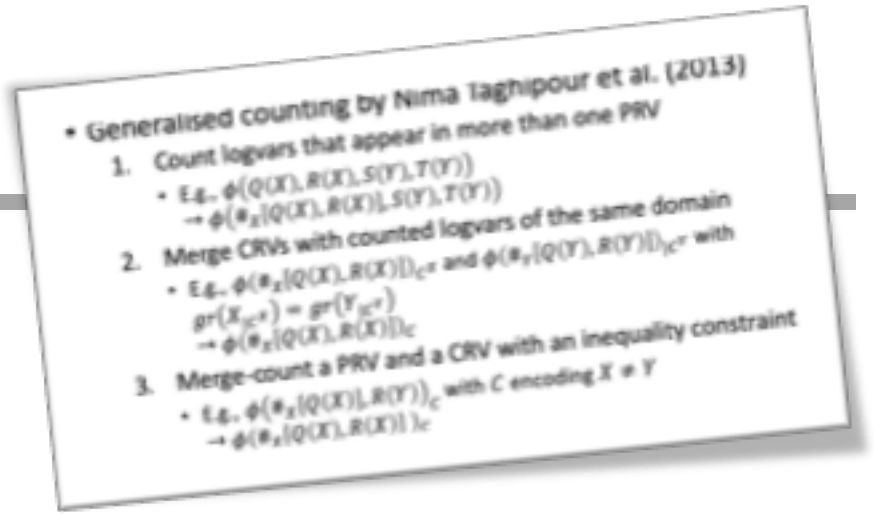
$$g\big(A(X,Y),B(X,Y)\big)$$
$$g(A(X,Y),C(X),C(Y)),X \neq Y$$
$$g(A(X,Y),D(X),E(Y))$$

  - $\mathcal{M}^{1prv}$: One logvar per PRV (arbitrarily many logvars per parfactor)

$$g\big(A(X),B(Y),C(Z)\big)$$

  - Holds for various lifted algorithms, e.g., LVE, LJT, FOKC

Completeness



- LVE is complete for $\mathcal{M}^{1prv}$ if considering all generalised counting versions
  - $\mathcal{M}^{1prv}$: One logvar per PRV
- Proof:
  - Fact: Only 1-logvar PRVs to eliminate
  1. Perform count conversion on all logvars in the model, possible scenarios in each parfactor
     - A. Logvar is the only one with a particular domain → Standard count conversion applies
     - B. Logvar occurs in several PRVs without inequality constraints → Generalised Counting 1 applies
     - C. Logvar occurs in several PRVs with inequality constraints → After counting logvar in those PRVs where Scenario B applies, Generalised Counting 3 applies
     - Afterwards: No uncounted logvars remain
  2. Multiply all parfactors into one large parfactor and merge CRVs with counted logvars of same domain (Generalised Counting 2)
  3. Eliminate all merged CRVs
     - Possible since the different CRVs do not overlap after Step 2
  4. Eliminate all propositional random variables

Completeness

- LVE is complete for $\mathcal{M}^{2lv}$
  - $\mathcal{M}^{2lv}$: Maximum of two logical variables per parfactor
- Requires another operator: Group Inversion
  - For the case $\phi\big(F(X,Y),F(Y,X)\big)_{|C}$, C encodes $X \neq Y$
    - Cannot sum out $F(X,Y)$ independently of $F(Y,X)$ as they refer to the same set of grounded random variables
  - Sums out PRVs $\{A_1,\ldots,A_k\}$ from $\phi(\mathcal{A})_{|C}$ at once where
    - $lv(A_1) = \cdots = lv(A_k) = lv(\mathcal{A})$
    - C encodes $X_i \neq X_j$ for each pair of logvars X_i, X_j, $\mathcal{D}(X_i) = \mathcal{D}\big(X_j\big)$

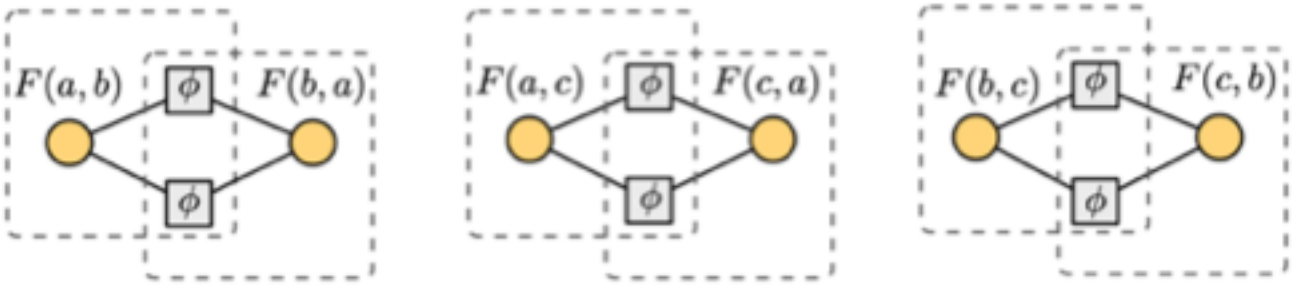


Figure taken from: Nima Taghipour: Lifted Probabilistic Inference by Variable Elimination. *PhD Thesis*, 2013.
Group Inversion: Nima Taghipour, Daan Fierens, Guy Van den Broeck, Jesse Davis, and Hendrik Blockeel: Completeness Results for Lifted Variable Elimination. In: *AISTATS-13 Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, 2013. (or Nima Taghipour's PhD thesis)

Completeness

- LVE is complete for $\mathcal{M}^{2lv}$
  - $\mathcal{M}^{2lv}$: Maximum of two logical variables per parfactor
- Proof idea:
  - Fact 1: Each parfactor has two logvars X, Y at most
  - Fact 2: Once two-logvar PRVs are eliminated, model is in $\mathcal{M}^{1prv}$
  1. Multiply all parfactors together that share two-logvar PRVs
     - Preserves the number of logvars per parfactor, namely, two
  2. Eliminate each two-logvar PRV in each parfactor; possible scenarios
     A. Only two-logvar PRVs in a parfactor with no inequality constraint → Eliminate using standard summing out
     B. Two-logvar PRVs with an inequality constraint → Eliminate using group inversion
     - Afterwards: Only one-logvar PRVs and propositional random variables remain (Fact 2)
  3. Count logvars in all parfactors, multiply the parfactors and merge CRVs, eliminate CRVs and propositional random variables (compare proof for completeness of $\mathcal{M}^{1prv}$)

Completeness

- Models with other constellations may be computed without groundings but not all possible models
  - E.g., for lifted variable elimination, models with three logical variables

$$g\big(A(X,Y,Z), B(X,Y), C(X)\big) \rightarrow liftable$$

$$g\big(F(X,Y), F(Y,Z), K(X,Z)\big) \rightarrow not\ liftable$$

→ Not complete for class $\mathcal{M}^{3lv}$, i.e., models with three logvars per parfactor

Completeness Beyond Models

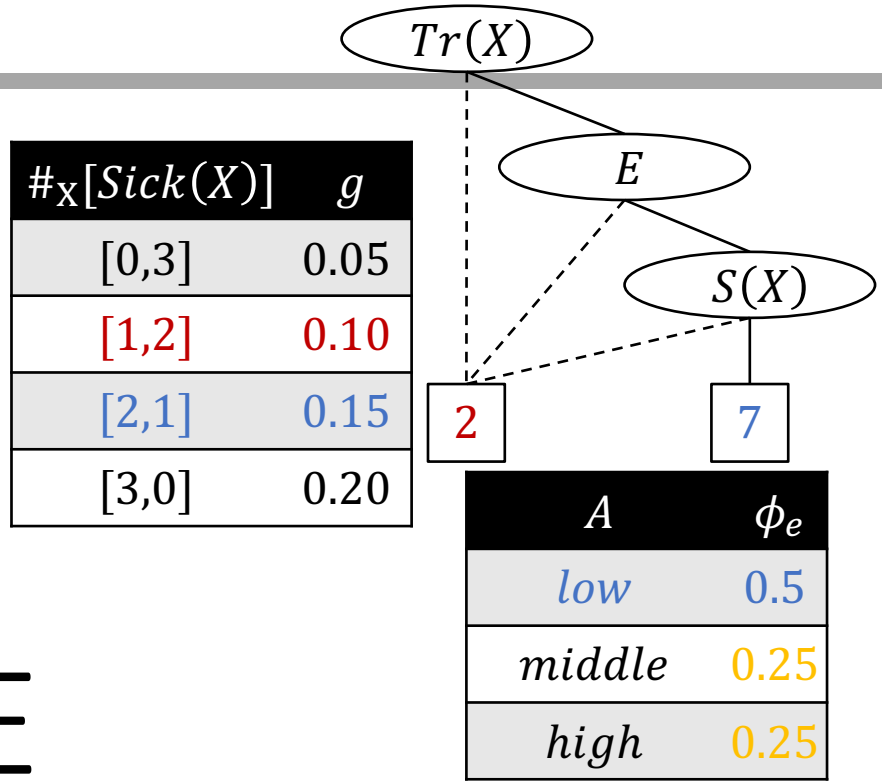
- Completeness results assumed a liftable class of queries $\mathcal{Q}$ and a liftable class of evidence $\mathcal{E}$
  - Liftable class of queries $\mathcal{Q}$:
    class of one ground query term
    - As argued on earlier slide, one query term does not influence complexity and cannot cause groundings
  - Liftable class of evidence $\mathcal{E}$:
    class of evidence on propositional random variables and one-logvar PRVs
    - General evidence on two-logvar PRVs no longer liftable in all cases
      - Lifted calculations possible for some cases but not for all
      - Proof by reduction to #2SAT problem

Guy Van den Broeck and Jesse Davis: Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference. In: *AAAI-12 Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2012.

Interim Summary

- FO dtrees
  - Cutset, context, cluster → lifted width
  - Liftable models
- Complexity
  - No longer exponential in domain sizes given liftable model → tractability
- Completeness
  - No groundings for
    - Models with two logvars per parfactor
    - Models with one-logvar PRVs and propositional random variables
  - Liftable query terms, liftable evidence

Beyond Standard LVE



$\#_X[Sick(X)]$	g
[0,3]	0.05
[1,2]	0.10
[2,1]	0.15
[3,0]	0.20

A	ϕ_e
low	0.5
middle	0.25
high	0.25

More query terms, uncertain evidence, function encoding

(Moves away from Nima Taghipour's PhD thesis)

More Query Terms

- So-called conjunctive query
  - $P(\boldsymbol{S}|\boldsymbol{T})$ with $|\boldsymbol{S}| > 1$
  - Inference task: Solve an instance of the query answering problem
    - Compute an answer to a query $P(\boldsymbol{S}|\boldsymbol{T})$ given a model G representing the full joint probability distribution P_G
    - Query for a marginal (conditional) probability (distribution)
    - Avoid grounding (parts of) G
- Does not change LVE in any major way
  - Now: Shatter on $\boldsymbol{S}$, Eliminate everything that is not $\boldsymbol{S}$
    - Instead of: Shatter on S, eliminate everything that is not S
  - E.g., $P\big(Travel(eve), Sick(eve)\big)$
    - Shatter on $Travel(eve), Sick(eve)$
    - Eliminate every term that is not $Travel(eve), Sick(eve)$

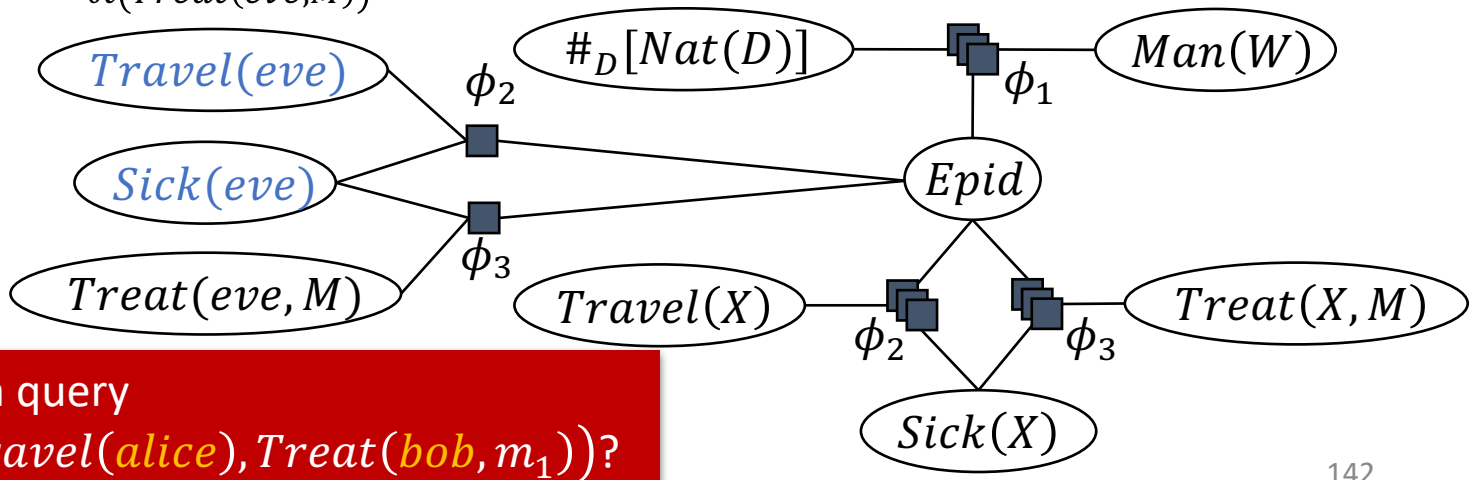
LVE: Example – Complete Derivation

$$P\big(Travel(eve), Sick(eve)\big)$$

$$= \frac{1}{Z} \sum_{e \in \mathcal{R}(Epid)} \phi_2^e\big(Travel(eve), e, Sick(eve)\big) \sum_{h_n \in \mathcal{R}(\#_D[Nat(D)])} Mul(h_n) \left(\sum_{m \in \mathcal{R}(Man(W))} \phi_1(e, m, h_n) \right)^{|\mathcal{D}(W)|}$$

$$\left(\sum_{s \in \mathcal{R}(Sick(X))} \left(\sum_{tt \in \mathcal{R}(Treat(X,M))} \phi_3(e, s, tt) \right)^{|\mathcal{D}(M)|} \sum_{t \in \mathcal{R}(Travel(X))} \phi_2(e, s, t) \right)^{|\mathcal{D}(X)|}$$

$$\sum_{te \in \mathcal{R}(Treat(eve,M))} \phi_3(e, Sick(eve), te)$$



What if we have a query $P\big(Sick(eve), Travel(alice), Treat(bob, m_1)\big)$?

More Query Terms

- Allowing conjunctive queries
  - $P(\boldsymbol{S}|\boldsymbol{T})$ with $|\boldsymbol{S}| > 1$
- Changes liftability, complexity, and completeness considerations
  - $P\big(Sick(eve), Travel(alice), Treat(bob, m_1)\big)$
    - Effectively grounds X if $\mathcal{D}(X) = \{alice, eve, bob\}$
- So far, query terms did not have an effect on runtime complexity and therefore, liftability and completeness
- With conjunctive queries, need to consider combined complexity of model and query

(Liftable) Conjunctive Query Classes

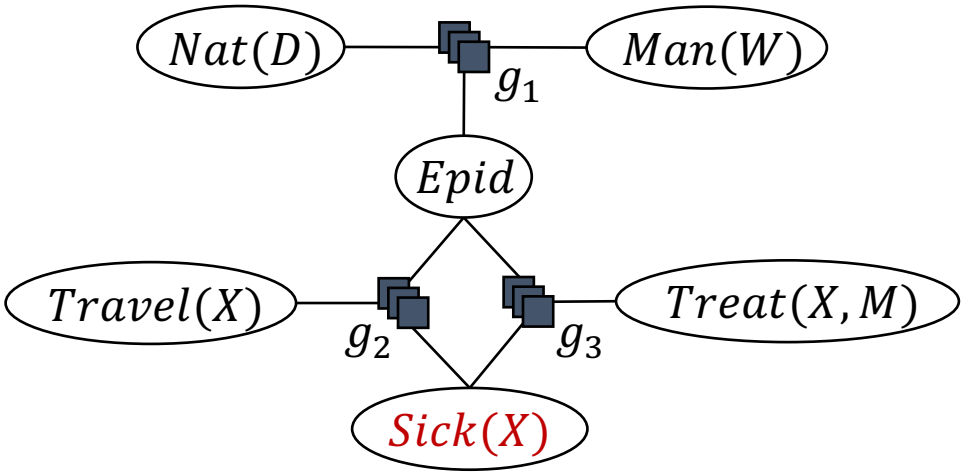
- $\mathcal{CQ}$: Class of all ground conjunctive queries $\boldsymbol{Q}$ given a model class with at least one logvar per model
  - LVE is not complete for all queries in $\mathcal{CQ}$ given a liftable model, liftable evidence
    - Proof by counter example (as on previous slide):
      - $P\big(Sick(eve), Travel(alice), Treat(bob, m_1)\big)$
      - Grounds X
      - LVE no longer polynomial in domain size
        - On-demand shattering only delays the problem
- $\mathcal{CQ}^{lift}$: Class of query terms $\boldsymbol{Q}$ containing at most one constant for each logvar in $lv(G)$
  - LVE is complete
    - Argument: Splits do not lead to a set of parfactors whose size depends on the domain size of logvars
  - Queries out of $\mathcal{CQ}^{lift}$ are liftable

Complexity

- Given query terms $\boldsymbol{Q}$ from $\mathcal{CQ}^{lift}$
  - Class of query terms $\boldsymbol{Q}$ containing at most one constant for each logvar in $lv(G)$
- Assumption is still that $q = |\boldsymbol{Q}|$ is reasonably small
  - Especially if comparing r^q to $r^{w_g} \cdot {n_\#}^{r_\# w_\#}$
  - S.t. we can consider it outweighed by $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$
- I.e., LVE complexity given a liftable model, a liftable query, and liftable evidence remains at

$$O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$$

Are those the only liftable queries?



- Consider

$$P\big(Sick(eve), Sick(alice), Sick(bob)\big)$$

  - Different constants for the same logvar
  - But: Same PRV

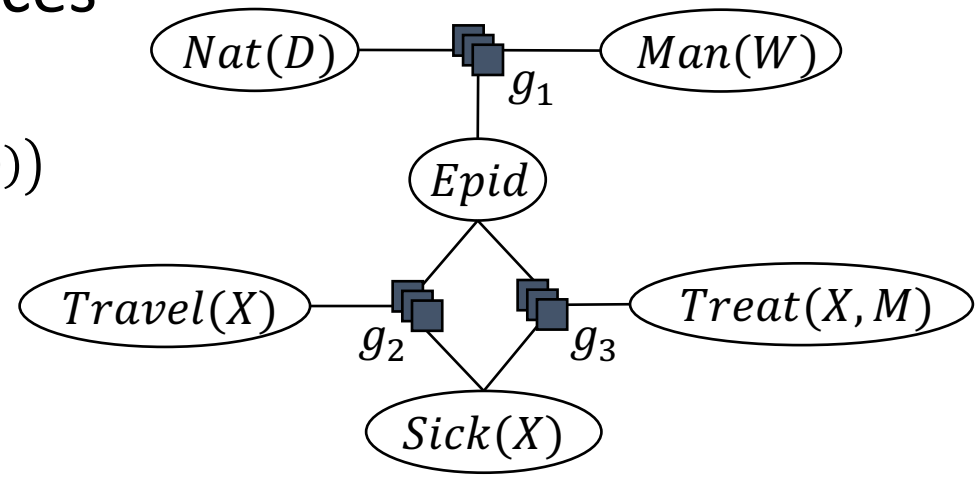
Tanya Braun and Ralf Möller: Parameterised Queries and Lifted Query Answering. In: *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

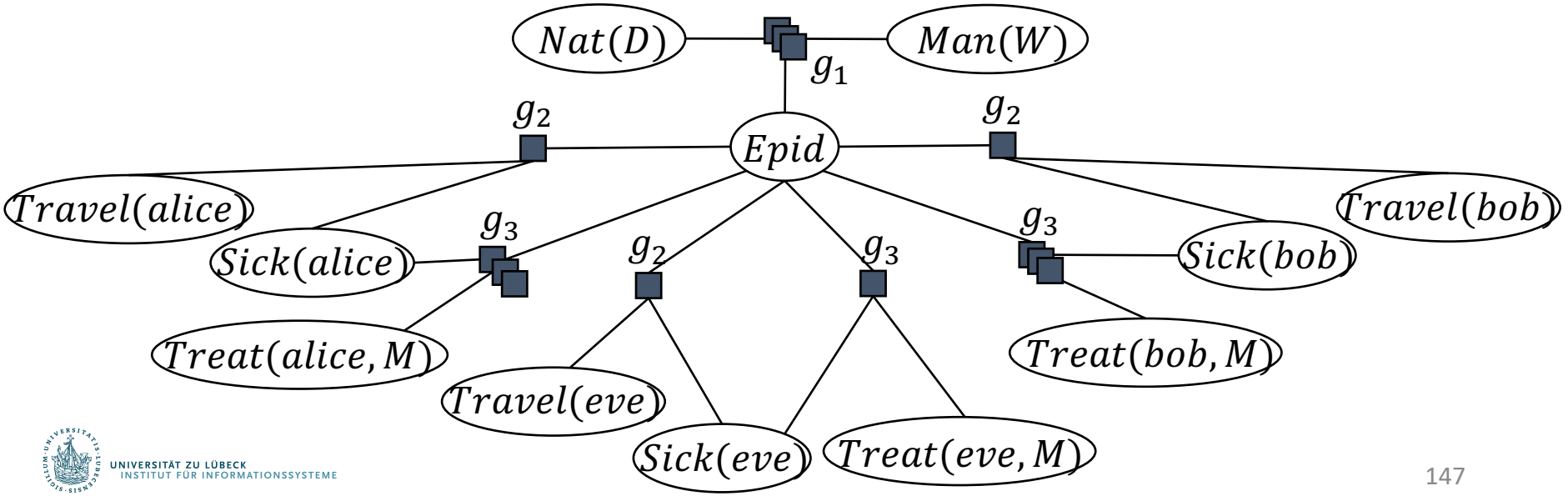
Indistinguishable Query Terms

- Indistinguishable instances in query:

  $P\big(Sick(alice), Sick(eve), Sick(bob)\big)$

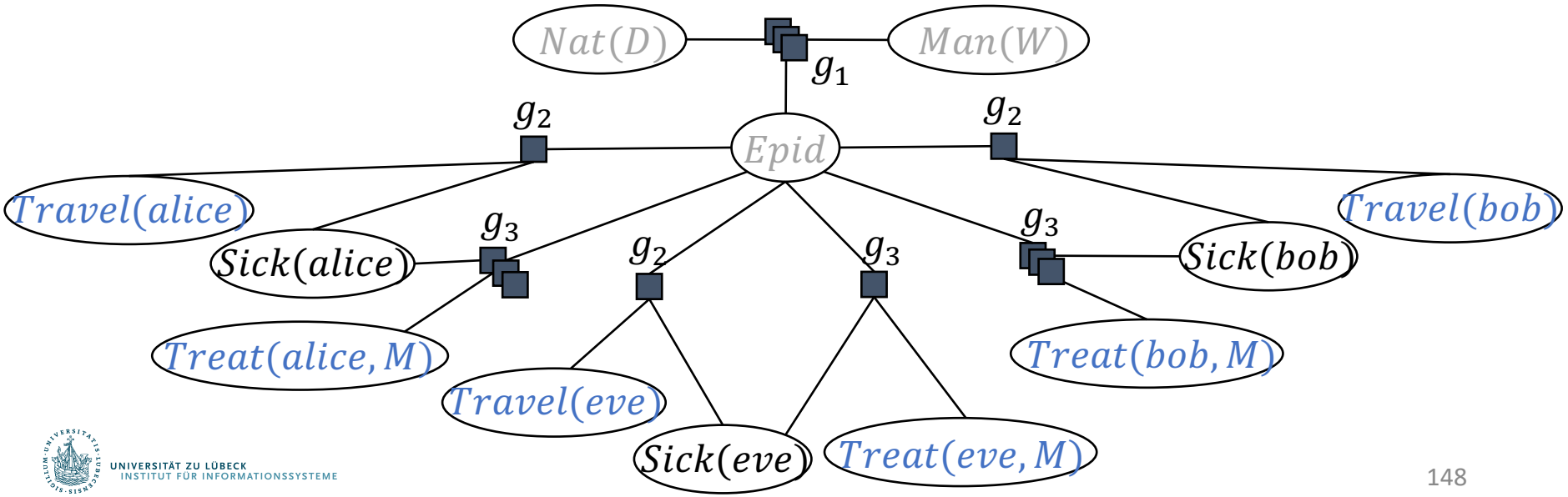
- Standard LVE:
  - Shattering
    - Leads to groundings w.r.t. constants in query





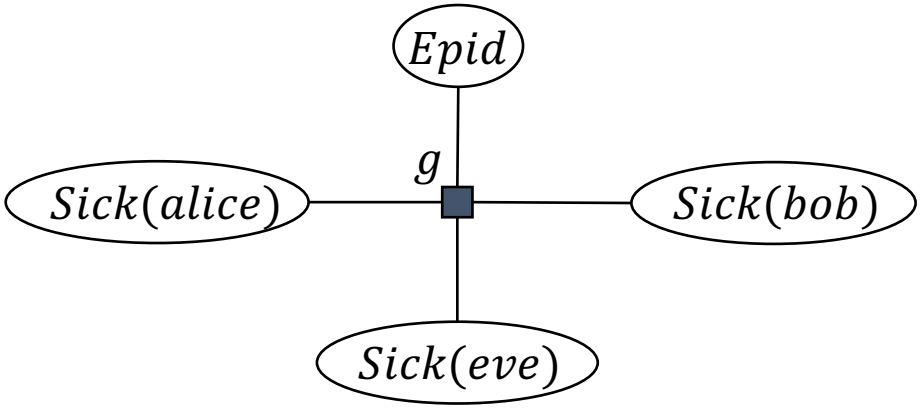
… And Their Effect

- Query: $P\big(Sick(alice), Sick(eve), Sick(bob)\big)$
- After shattering, eliminate all non-query terms
  - Identical computations during elimination



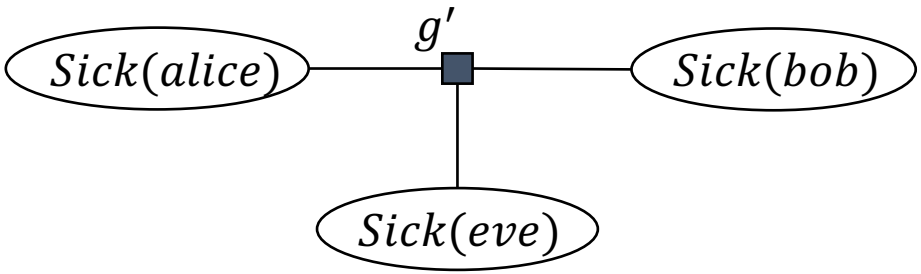
… And Their Effect

- Query: $P\big(Sick(alice), Sick(eve), Sick(bob)\big)$
- After shattering, eliminate all non-query terms
  - Identical computations during elimination
  - Large intermediate results



… And Their Effect

- Query: $P\big(Sick(alice), Sick(eve), Sick(bob)\big)$
- After shattering, eliminate all non-query terms
- Symmetries in result
  - Encode with CRV



$\#_X[Sick(X)]$	g
[0,3]	1
[1,2]	2
[2,1]	3
[3,0]	4

$Sick(alice)$	$Sick(eve)$	$Sick(bob)$	g'
$false$	$false$	$false$	1
$false$	$false$	$true$	2
$false$	$true$	$false$	2
$false$	$true$	$true$	3
$true$	$false$	$false$	2
$true$	$false$	$true$	3
$true$	$true$	$false$	3
$true$	$true$	$true$	4

Parameterised Queries

- Allow logvars in query
- Definition of a query: $P(\boldsymbol{S}_{|C}|\boldsymbol{T})$
  - C a constraint on the logvars of $\boldsymbol{S}$
    - $C = \top$
      - If no restriction on domain constants apply
      - Can be omitted
  - Represents a conjunctive query $P(gr(\boldsymbol{S}_{|C})|\boldsymbol{T})$
  - Still allows for queries as seen so far
- E.g., $P(Sick(X)_{|\top})$
  - represents $P(Sick(alice), Sick(eve), Sick(bob))$
- At the end, the result contains the logvars counted
  - If counting the logvars of $\boldsymbol{S}_{|C}$ is not possible, then LVE needs to ground them to ensure a distribution over $\boldsymbol{S}_{|C}$

Tanya Braun and Ralf Möller: Parameterised Queries and Lifted Query Answering. In: *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

LVE for Parameterised Queries

LVE$(G, \boldsymbol{Q}_{|C}, \{g_e\}_{e=1}^m)$

$G \leftarrow$ Shatter G on $\boldsymbol{Q}_{|C}$, $\{g_e\}_{e=1}^m$, and on itself

$G \leftarrow$ Absorb $\{g_e\}_{e=1}^m$ in G

while G contains non-query terms **do**

if a PRV A fulfils the preconditions of $\mathrm{sum{-}out}$ **then**

$G \leftarrow$ Apply $\mathrm{sum{-}out}$ to A in G

else

$G \leftarrow$ Apply an enabling operator ($\mathrm{multiply}$, $\mathrm{count{-}convert}$, expand, $\mathrm{count{-}normalise}$, split, ground) on some parfactors in G

while $lv(G) \neq \emptyset$ **do**

if $\exists X \in lv(G)$ s.t. X is countable in $g \in G$ **then**

$G \leftarrow$ Apply $\mathrm{count{-}convert}$ to X in g

else

$G \leftarrow$ Apply an enabling operator ($\mathrm{multiply}$, expand, $\mathrm{count{-}normalise}$, split, ground) on some parfactors in G

$g \leftarrow$ Multiply all parfactors in G into one parfactor

$g \leftarrow$ *Normalise* the potentials in g

return g

At this point, G contains only $\boldsymbol{Q}_{|C}$ terms but the logvars in $\boldsymbol{Q}_{|C}$ may still be uncounted; the next loop counts them if possible

Normalisation changes because of the histograms representing multiple assignments

Tanya Braun and Ralf Möller: Parameterised Queries and Lifted Query Answering. In: *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

Normalisation

- Histograms encode multiple assignments
  - $Mul(h)$ assignments have the potential $\phi(h)$
  - Incorporate into normalisation
- To get the probability of one assignment $\boldsymbol{a}$ behind h in parfactor $\phi(\#_X[R(X)])$:

$$p_{\boldsymbol{a}|h} = \frac{\phi(h)}{\sum_{h \in \mathcal{R}(\#_X[R(X)])} Mul(h) \cdot \phi(h)}$$

  - Probability of $Mul(h)$ assignments

$$p_h = Mul(h) \cdot p_{\boldsymbol{a}|h}$$

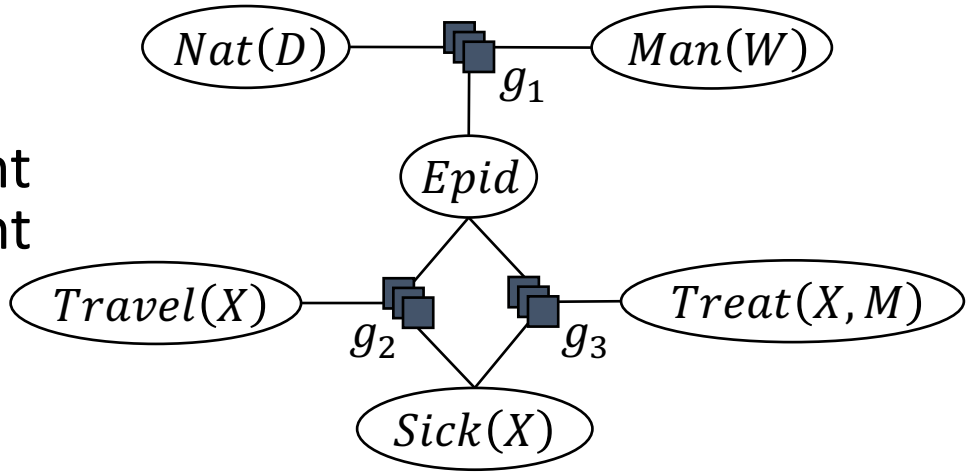
  - Distribution:

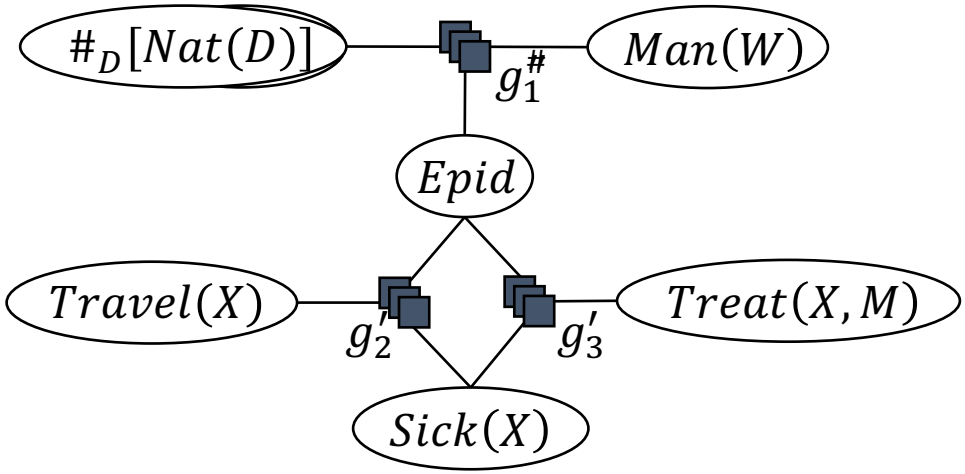
$$\sum_{h \in \mathcal{R}(\#_X[R(X)])} p_h = 1$$

$\#_X[Sick(X)]$	g
[0,3]	1
[1,2]	2
[2,1]	3
[3,0]	4

Example

- Query: $P\big(Sick(X)\big)$
- No shattering
  - No effect since constraint of X in query = constraint of X in model
- No evidence absorption
- Elimination:
  - Sum-out $Treat(X, M)$
  - Sum-out $Travel(X)$
  - Count-convert $Nat(D)$
  - Sum-out $Man(W)$
  - Sum-out $\#_D[Nat(D)]$

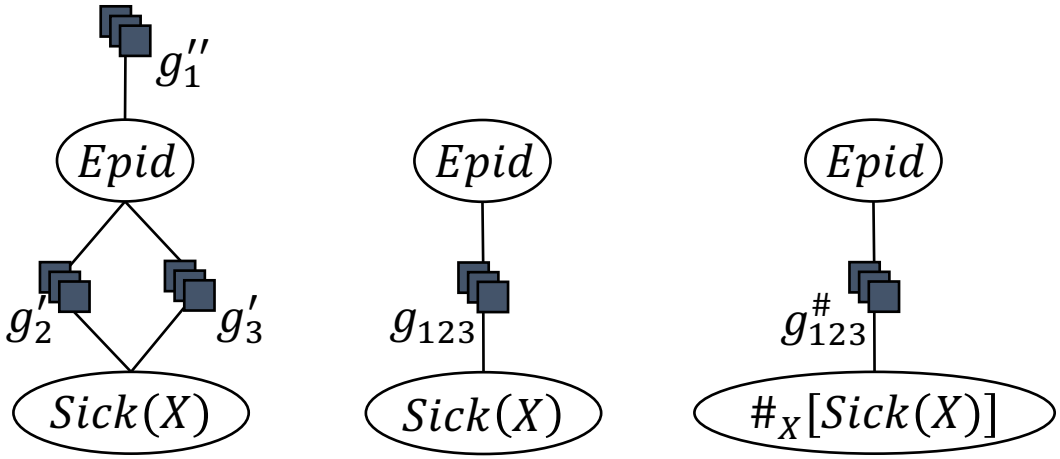




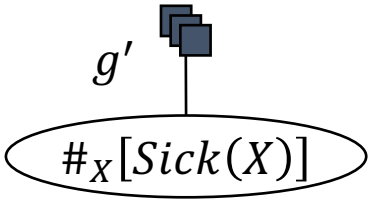
Example

- Query: $P(Sick(X))$
- Elimination:
  - Only random variable left to eliminate: $Epid$
    - Occurs in three parfactors: Multiply
    - Does not contain X: Count-convert X
    - Eliminate $Epid$

Here, count conversion as part of elimination, but if logvars remaining after elimination, count conversion also trivially possible



$\#_X[Sick(X)]$	g'
[0,3]	1
[1,2]	2
[2,1]	3
[3,0]	4



Example

- Query: $P\big(Sick(X)\big)$
- Elimination: Finished
- Normalisation:

$$p_{\boldsymbol{a}|h} = \frac{\phi(h)}{\sum_{h \in \mathcal{R}(\#_X[R(X)])} Mul(h) \cdot \phi(h)} = \frac{\phi(h)}{1 \cdot 1 + 3 \cdot 2 + 3 \cdot 3 + 1 \cdot 4} = \frac{\phi(h)}{20}$$

$\#_X[Sick(X)]$	g'	
[0,3]	1	$\rightarrow {}^{1}/_{20}$
[1,2]	2	$\rightarrow {}^{2}/_{20}$
[2,1]	3	$\rightarrow {}^{3}/_{20}$
[3,0]	4	$\rightarrow {}^{4}/_{20}$

- Distribution:

$$\sum_{h \in \mathcal{R}(\#_X[R(X)])} Mul(h) \cdot p_{\boldsymbol{a}|h}$$

$$= 1 \cdot \frac{1}{20} + 3 \cdot \frac{2}{20} + 3 \cdot \frac{3}{20} + 1 \cdot \frac{4}{20} = \frac{1+6+9+4}{20} = \frac{20}{20} = 1$$

Splits Affecting Query Logvars

- Logvars $\boldsymbol{X}$ in query terms may be split in result
  - If splits of the model affect the query logvars
    - Including groundings
- Prominent case: evidence; three cases given a query term $R(\boldsymbol{X})_{|C}$ and evidence PRV $E(\boldsymbol{Y})_{|C_e}$
  1. Overlap between instances $gr\left(R(\boldsymbol{X})_{|C}\right) \cap gr\left(E(\boldsymbol{X})_{|C_e}\right) \neq \emptyset$ (i.e., $R(\boldsymbol{X}) = E(\boldsymbol{Y})$)
     - Split C on the overlap with C_e, i.e., $C/_{\boldsymbol{X}}C_e$
       - Instances of C_e will be absorbed
     - Answer has $R(\boldsymbol{X})_{|C}$ partitioned into $C/_{\boldsymbol{X}}C_e \setminus C_e$ in the result
     - E.g., $P\left(Sick(X)_{|(X,\{x_i\}_{i=1}^{20})}\right)$ and $\phi_e\left(Sick(X)\right)_{|(X,\{x_i\}_{i=1}^{10})}$
       - Result: $\phi(\#_X[Sick(X)])_{|(X,\{x_i\}_{i=11}^{20})}$

Splits Affecting Query Logvars

- Logvars $\boldsymbol{X}$ in query terms may be split in result
  - If splits of the model affect the query logvars
    - Including groundings
- Prominent case: evidence; three cases given a query term $R(\boldsymbol{X})_{|C}$ and evidence PRV $E(\boldsymbol{Y})_{|C_e}$
  2. Overlap of constants (but not instances)
     $gr\left(\pi_{\boldsymbol{Z}}\left(\boldsymbol{X}_{|C}\right)\right) \cap gr\left(\pi_{\boldsymbol{Z}}\left(\boldsymbol{Y}_{|C_e}\right)\right) \neq \emptyset$ ($\boldsymbol{Z}$ shared logvars)
     - Split C on the overlap with C_e, i.e., $C/_{\boldsymbol{Z}}C_e$
     - Answer has $R(\boldsymbol{X})_{|C}$ partitioned into $C/_{\boldsymbol{X}}C_e$ in the result
     - E.g., $P\left(Sick(X)_{|(X,\{x_i\}_{i=1}^{20})}\right)$ and $\phi_e\left(Travel(X)\right)_{|(X,\{x_i\}_{i=1}^{10})}$
       - Result: $\phi(\#_{X'}[Sick(X')], \#_{X''}[Sick(X'')])_{|\left((X',X''),\{x_i\}_{i=1}^{10}\times\{x_i\}_{i=11}^{20}\right)}$

Splits Affecting Query Logvars

- Logvars $\boldsymbol{X}$ in query terms may be split in result
  - If splits of the model affect the query logvars
    - Including groundings
- Prominent case: evidence; three cases given a query term $R(\boldsymbol{X})_{|C}$ and evidence PRV $E(\boldsymbol{Y})_{|C_e}$
  3. No overlap (more of a non-case)

$$gr\left(R(\boldsymbol{X})_{|C}\right) \cap gr\left(E(\boldsymbol{Y})_{|C_e}\right) = \emptyset$$
$$\wedge\, no\ shared\ logvars\ \boldsymbol{Z}$$

  - $R(\boldsymbol{X})_{|C}$ is not partitioned in the result because of evidence
  - E.g., $P\left(Sick(X)_{|(X,\{x_i\}_{i=1}^{20})}\right)$ and $\phi_e\left(Nat(D)\right)_{|(D,\{d_i\}_{i=1}^{2})}$
    - Result: $\phi(\#_X[Sick(X)]\,)_{|(X,\{x_i\}_{i=1}^{20})}$

Completeness

- Given a liftable model and liftable evidence
- $\mathcal{PCQ}$: class of all parameterised conjunctive queries
  - LVE is incomplete
    - Proof by counter example, using constraints:
      - $P\big(Sick(X'), Travel(X'')\big)_{|\big((X',X''),\{alice,eve\}\times\{eve,bob\}\big)}$
    - Or using logvars:
      - Query $P\big(B(X,Y)\big)$ in model $g\big(A(X), B(X,Y), C(Y)\big)$
- $\mathcal{PCQ}^{lift}$: Parameterised query terms with only one parameter per term and one subset of constants per domain
  - LVE is complete
    - Proof along the lines of proving completeness for $\mathcal{M}^{1prv}$
  - Queries in $\mathcal{PCQ}^{lift}$ are called liftable
- Corollary
  - CRVs compactly represent the result of liftable queries

Complexity

- Given a liftable model, liftable evidence, and a liftable query
- Runtime complexity of LVE for liftable queries does not change
  - I.e., remains
$$O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$$
  - Argument: Given a liftable query, only $\mathrm{count}-\mathrm{convert}$ operations are additionally applied, which do not depend on domain sizes and have a complexity of $O(\log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$

Uncertain Evidence

- Assumption about evidence: Certainty!
  - Evidence assumed to be correct
    - Encoded with a 1 for the observed value and 0's elsewhere
    - E.g., $Sick(x_1) = Sick(x_2) = \cdots = Sick(x_{10}) = true$
      - Parfactor $g_e^T = \phi_e^T\big(Sick(X)\big)_{((X),\{x_1,\ldots,x_{10}\})}$

$Sick(X)$	ϕ_e^T
$false$	0
$true$	1

- Problem: Noisy or faulty observations

- Solution: Specify a prior beyond (1,0)-distributions

Marcel Gehrke, Tanya B, and Ralf Möller: Uncertain Evidence for Probabilistic Relational Models. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019.

Uncertain Evidence

- Associate observations with a probability (distribution) to account for noise in observations
  - Either specify a distribution over all range values
    - E.g.,
      - Boolean PRV $Sick(X)$
        - $Sick(X)_{|((X),\{x_1,\ldots,x_{10}\})} = false$ with $p_F = 0.2$
        - $Sick(X)_{|((X),\{x_1,\ldots,x_{10}\})} = true$ with $p_T = 0.8$
        - Parfactor $g_e = \phi_e\big(Sick(X)\big)_{|((X),\{x_1,\ldots,x_{10}\})}$

$Sick(X)$	ϕ_e
$false$	0.2
$true$	0.8

      - PRV A with range values $\{low, middle, high\}$
        - $A = low$ with $p_{low} = 0.5$
        - $A = middle$ with $p_{middle} = 0.3$
        - $A = high$ with $p_{high} = 0.2$
        - Parfactor $g_e^A = \phi_e^A(A)_{|\mathsf{T}}$

A	ϕ_e
low	0.5
$middle$	0.3
$high$	0.2

Marcel Gehrke, Tanya B, and Ralf Möller: Uncertain Evidence for Probabilistic Relational Models. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019.

Uncertain Evidence

- Associate observations with a probability (distribution) to account for noise in observations
  - Or specify probabilities $p_1, \dots, p_k$ for certain range values $r_1, \dots, r_k$ and distribute the remaining probability mass onto the remaining range values $r_{k+1}, \dots, r_n$

Syntactic sugar

    - Given $\{p_1, \dots, p_k\}$ with $\sum_{i=1}^{k} p_i \leq 1$ for $\{r_1, \dots, r_k\}$ of overall range values $\{r_1, \dots, r_k, r_{k+1}, \dots, r_n\}$, probability p_j for $j \in \{k+1, \dots, n\}$:

$$p_j = \frac{1}{n-k}\left(1 - \sum_{i=1}^{k} p_i\right)$$

Remaining probability mass

    - E.g.,
      - $Sick(X)_{|((X),\{x_1,\dots,x_{10}\})} = true$ with $p = 0.8$
      - $A = low$ with $p_{low} = 0.5$

$Sick(X)$	ϕ_e
$false$	0.2
$true$	0.8

A	ϕ_e
low	0.5
$middle$	0.25
$high$	0.25

Marcel Gehrke, Tanya B, and Ralf Möller: Uncertain Evidence for Probabilistic Relational Models. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019.

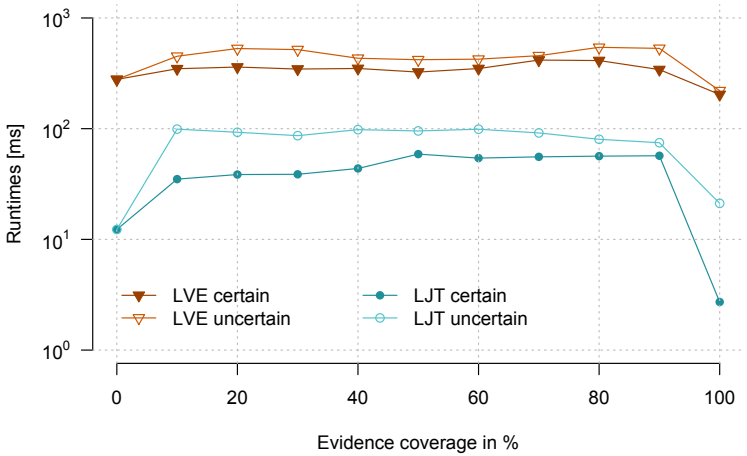
Uncertain Evidence

- As prior distributions, evidence parfactors become part of the model
  - Related to Pearl's method of virtual evidence (1988)
- There is research on reworking model distributions to also have the a posteriori distribution be identical to the given prior distribution
  - Inspired by the fact that with certain evidence, the a priori and a posteriori distributions are both (1,0)-distributions
  - Also called soft evidence, proposed by Jeffrey (1990)
- Translation of one into the other is possible (Chan and Darwiche, 2005)

Judea Pearl: Probabilistic reasoning in intelligent systems: Networks of plausible reasoning. 1988.
R.C. Jeffrey: The Logic of Decision. University of Chicago Press, 1990.
Hei Chan and Adnan Darwiche: On the revision of probabilistic beliefs using uncertain evidence. In: *Artificial Intelligence*, 2005.

LVE with Uncertain Evidence

- No fundamental changes
  - Absorption only applies to evidence parfactors with (1,0)-distributions
  - With other distributions, the evidence parfactor is added to the model and handled like any other model parfactor
    - No dimension reduction in this case
- Runtime will increase minimally compared to certain evidence
  - E.g., epidemic example
    - Evidence for 0% to 100% on one-logvar PRVs
      - Domain size 1000
    - (Ignore LJT in figure for now)



Marcel Gehrke, Tanya B, and Ralf Möller: Uncertain Evidence for Probabilistic Relational Models. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019.

Liftability, Complexity, Completeness

- Liftability
  - Under the assumption that for each observed value o of a PRV A, there exists only one distribution
    - Liftability results still hold
      - Same amount of evidence parfactors as before
    - If we allow different distributions for each observed INSTANCE (not observed value), then each instance gets an own parfactor and the model is grounded during shattering
- Complexity
  - Given liftable uncertain evidence, complexity does not change as LVE remains the same
- Completeness
  - Complete for evidence on one-logvar PRVs as before with above assumption on distributions per observed value

Marcel Gehrke, Tanya B, and Ralf Möller: Uncertain Evidence for Probabilistic Relational Models. In: *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019.

Encoding Functions (Idea)

- Potential functions can have local symmetries
  - I.e., potentials that a potential function maps to multiple times
  - As seen when translating MLNs into PMs
    - E.g., ϕ can be represented with two formulas

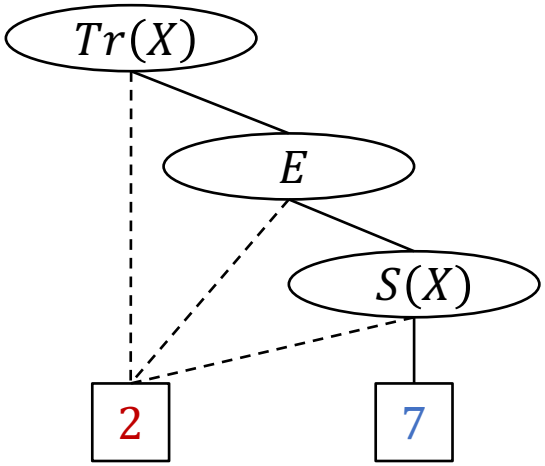
$$(\ln 2\,, \neg travel(X) \vee \neg epid \vee \neg sick(X))$$
$$(\ln 7\,, travel(X) \wedge epid \wedge sick(X))$$

$Travel(X)$	$Epid$	$Sick(X)$	ϕ
$false$	$false$	$false$	2
$false$	$false$	$true$	2
$false$	$true$	$false$	2
$false$	$true$	$true$	2
$true$	$false$	$false$	2
$true$	$false$	$true$	2
$true$	$true$	$false$	2
$true$	$true$	$true$	7

- So is there a different encoding of potential functions than lists that are useful?

Algebraic Decision Diagrams (ADDs)

- Represent the function as a binary tree
  - Requires Boolean range values
    - Multi-valued variants of ADDS exist!
  - Potentials are the leaves
  - Inner nodes labelled with PRV
    - Left child: assignment of $false$
    - Right child: assignment of $true$



$Travel(X)$	$Epid$	$Sick(X)$	ϕ
$false$	$false$	$false$	2
$false$	$false$	$true$	2
$false$	$true$	$false$	2
$false$	$true$	$true$	2
$true$	$false$	$false$	2
$true$	$false$	$true$	2
$true$	$true$	$false$	2
$true$	$true$	$true$	7

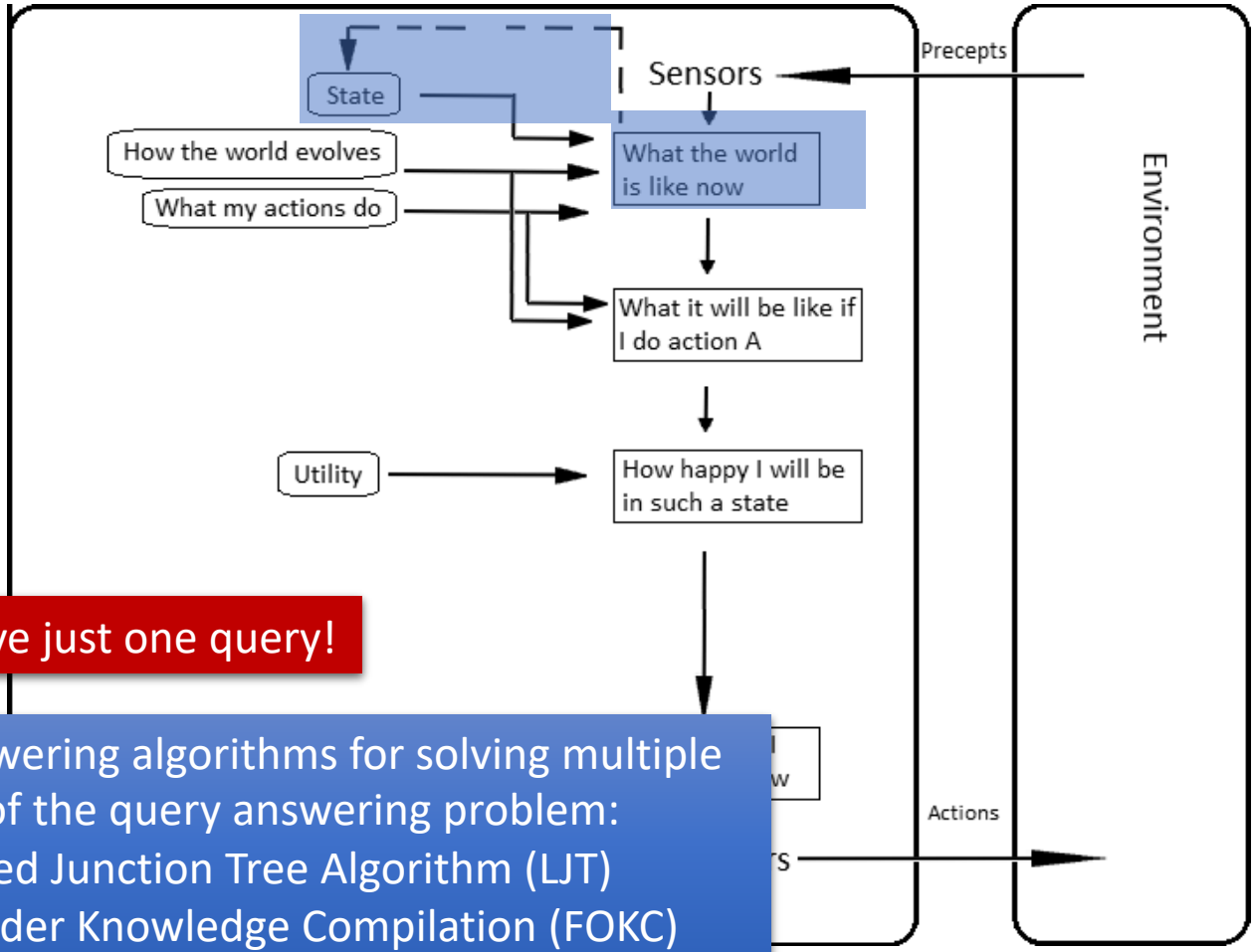
R. Iris Bahar, E.A. Frohm, Charles M. Gaona, G.D. Hachtel, E. Macii, Abelardo Pardo, and F. Somenzi: Algebraic Decision Diagrams and Their Application. In: *IEEE /ACM International Conference on CAD*, 1993.

ADDs: Up- and Downsides

- ADD may be much smaller than table specification of potential function with local symmetries
  - Worst case: same space complexity
- Multiplication and addition (main arithmetic operations during (L)VE) implementable on ADDs
  - If ADD much smaller than table, then fewer arithmetic operations to carry out overall
- But: overhead in handling ADDs
  - Build/update tree structure
- So: Only works with a lot of local symmetries

Mark Chavira and Adnan Darwiche: Compiling Bayesian Networks Using Variable Elimination. In: *IJCAI-07 Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007. (and in the book "Modelling and Reasoning with Bayesian Networks" by Adnan Darwiche, mentioned in Introduction; Chapter 13.5)

Setting: Agent with Utilities



Unlikely to have just one query!

Query answering algorithms for solving multiple instances of the query answering problem:
Lifted Junction Tree Algorithm (LJT)
First-order Knowledge Compilation (FOKC)

Interim Summary

- Query types
  - Conjunctive queries
    - Query class of liftable conjunctive queries
  - Parameterised queries
    - CRVs to encode symmetries in result
    - Query class of liftable parameterised queries
- Uncertain evidence
  - Probability distribution over observable values added as another parfactor
  - Assuming one distribution per evidence parfactor, no change in complexity, completeness results
- ADD encoding (idea)
  - Use ADD to compactly encode potential function with local symmetries
  - Overhead for managing ADDs

Outline: 3. Lifted Inference

A. *Lifted variable elimination (LVE)*
- Operators
- Algorithm
- Complexity (including first-order dtrees), completeness, tractability
- Variants

B. *Lifted junction tree algorithm (LJT)*
- First-order junction trees (FO jtrees)
- Algorithm
- Complexity, completeness
- Variants

C. *First-order knowledge compilation (FOKC)*
- Normal form, circuits
- Algorithm
- Complexity, completeness

D. *Most probable assignment queries*
- Distribution vs. assignment queries
- Most probable explanation (MPE) , Maximum-a-posteriori (MAP) assignments
- Changes in LVE, LJT, FOKC
- Complexity, completeness