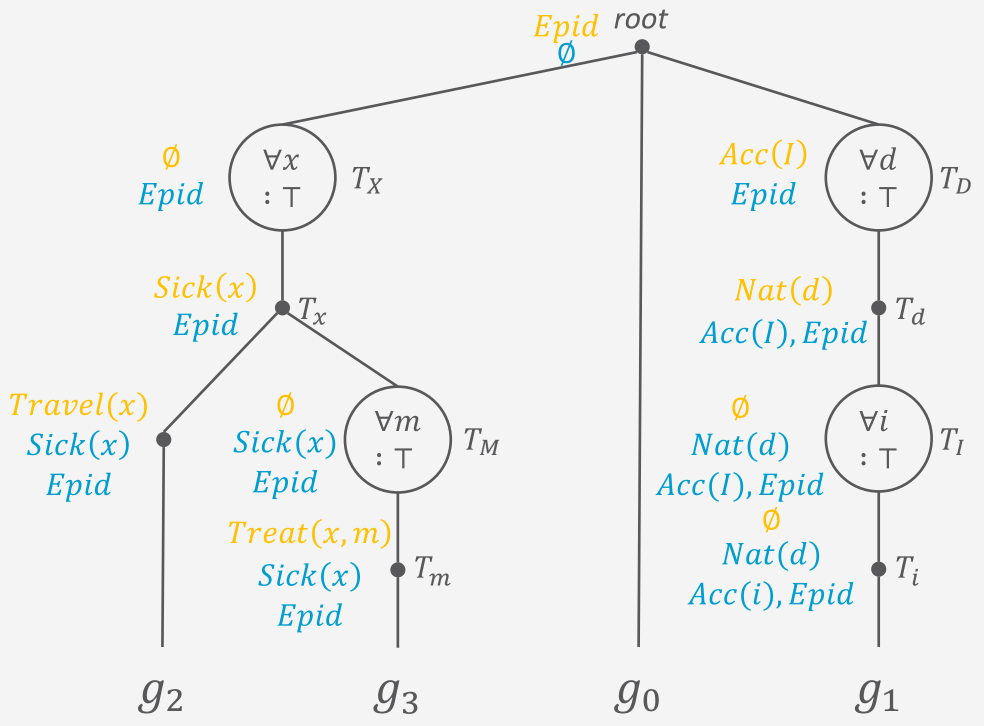
Lifted Inference: Exact Inference

Statistical Relational Artificial Intelligence (StaRAI)



Tanya Braun

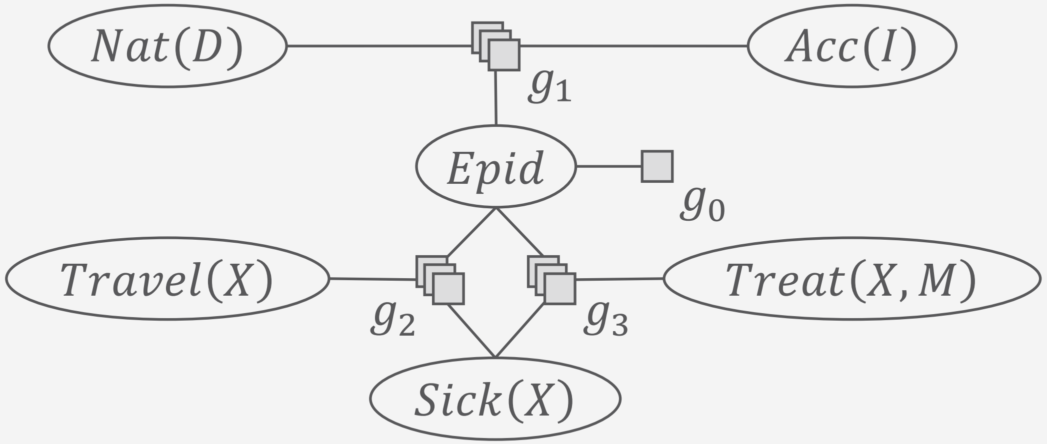
Contents

Inference Tasks

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

$10 \; Presents(X, P, C) \Rightarrow Attends(X, C)$

$3.75 \; Publishes(X, C) \wedge FarAway(C) \Rightarrow Attends(X, C)$



Outline: 4. Lifted Inference

A. Exact Inference

i. Lifted Variable Elimination for Parfactor Models
  - Idea, operators, algorithm, complexity

ii. Lifted Junction Tree Algorithm
  - Idea, helper structure: junction tree, algorithm

iii. First-order Knowledge Compilation for MLNs
  - Idea, helper structure: circuit, algorithm

B. Approximate Inference: Sampling

- Rejection sampling
- (Lifted) likelihood sampling
- (Lifted) Markov Chain Monte Carlo sampling

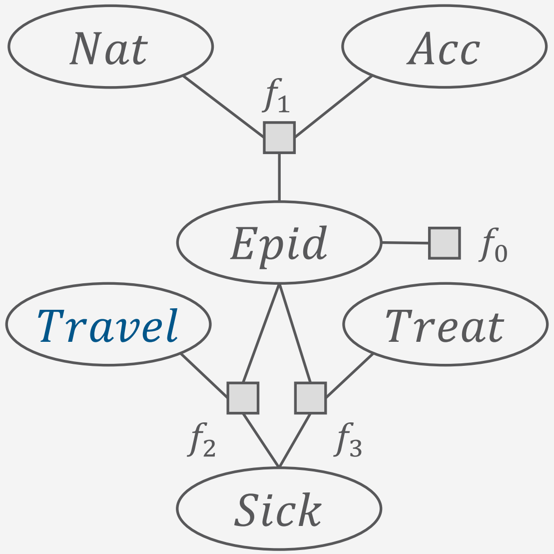
Remember: Variable Elimination (VE)

- Outline:
  1. Absorb evidence $\boldsymbol{t}$ in each factor covered by $\boldsymbol{t}$, i.e., $rv(f) \cap \boldsymbol{t} \neq \emptyset$,
  2. Sum out non-query variables $\boldsymbol{U} = \boldsymbol{R} \setminus rv(\boldsymbol{S}, \boldsymbol{t})$ using factorisation in model F

$$P(\boldsymbol{S} \mid \boldsymbol{t}) = \frac{1}{P(\boldsymbol{t})} \sum_{\boldsymbol{u} \in ran(\boldsymbol{U})} P_F(\boldsymbol{S}, \boldsymbol{t}, \boldsymbol{U} = \boldsymbol{u})$$

$$= \frac{1}{P(\boldsymbol{t})} \sum_{\boldsymbol{u} \in ran(\boldsymbol{U})} \prod_{f \in F} \phi_f \underbrace{(R_1, \dots, R_k)}_{\pi_{rv(f)}(\boldsymbol{S}, \boldsymbol{t}, \boldsymbol{U} = \boldsymbol{u})}$$

  - Factor out factors from sums if arguments not covered by sum
  3. Divide by $P(\boldsymbol{t})$ = Normalise $P(\boldsymbol{S}, \boldsymbol{t})$
- Example: $P(Travel)$ in $F = \{f_i\}_{i=0}^{3}$



Remember: Variable Elimination (VE): Example

$P(Travel)$

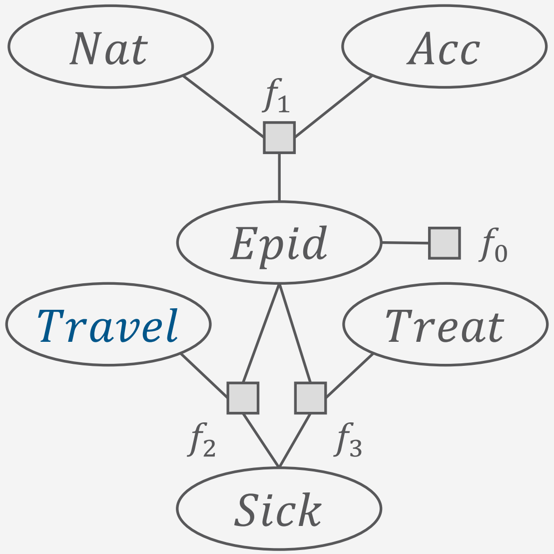
$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} P_{\boldsymbol{R}}(E=e, N=n, A=a, S=s, Travel, T=t)$$

$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} \prod_{i=0}^{3} \phi_i\,(\boldsymbol{R}_i = \boldsymbol{r}_i)$$

$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} \phi_0(e)\phi_1(e,n,a)\phi_2(Travel,e,s)\phi_3(e,s,t)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e,n,a) \sum_{s \in \text{Val}(S)} \phi_2(Travel,e,s) \sum_{t \in \text{Val}(T)} \phi_3(e,s,t)$$

Sums can be computed independently → could be done in parallel

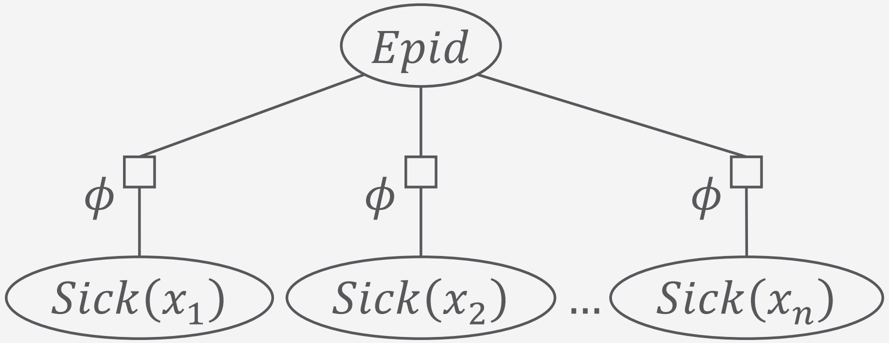


$E \triangleq Epid, N \triangleq Nat, A \triangleq Acc, S \triangleq Sick, T \triangleq Treat$

What Happens During Variable Elimination Given Relations?

$$P(Epid) \propto \sum_{s_1 \in ran(Sick(x_1))} \phi(Epid, s_1) \cdot \sum_{s_2 \in ran(Sick(x_2))} \phi(Epid, s_2) \cdot \dots \cdot \sum_{s_n \in ran(Sick(x_n))} \phi(Epid, s_n)$$

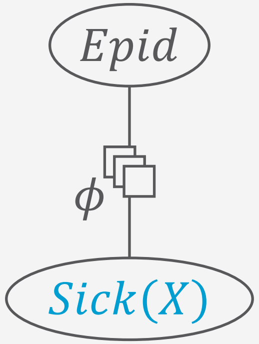
$$= \underbrace{\phi'(Epid) \cdot \phi'(Epid) \cdot \dots \cdot \phi'(Epid)}_{n \text{ times}} = \left(\phi'(Epid)\right)^n$$



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

\# of X relative to $Epid$

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



Lifted Variable Elimination (LVE)

- Outline:
  1. Absorb evidence $\boldsymbol{t}$ in each parfactor g covered by $\boldsymbol{t}$, i.e., $rv(g) \cap \boldsymbol{t} \neq \emptyset$, in a lifted way,
  2. Eliminate non-query PRVs $\boldsymbol{U} = \boldsymbol{R} \setminus rv(\boldsymbol{S}, \boldsymbol{t})$ in a lifted way in model G

$$P(\boldsymbol{S} \mid \boldsymbol{t}) = \frac{1}{P(\boldsymbol{t})} \sum_{\boldsymbol{u} \in ran(\boldsymbol{U})} P_G(\boldsymbol{S}, \boldsymbol{t}, \boldsymbol{U} = \boldsymbol{u})$$

$$= \frac{1}{P(\boldsymbol{t})} \sum_{\boldsymbol{u} \in ran(\boldsymbol{U})} \prod_{g \in G} \phi_g(\underbrace{R_1, \dots, R_k}_{\pi_{rv(f)}(\boldsymbol{S}, \boldsymbol{t}, \boldsymbol{U} = \boldsymbol{u})})$$

  - Factor out parfactors from sums if arguments not covered by sum
  - May require manipulation of constraints as at least constants appearing in query are *distinguishable*
  3. Divide by $P(\boldsymbol{t})$ = Normalise $P(\boldsymbol{S}, \boldsymbol{t})$

Lifted operators for
- Summing out
- Multiplication
- Absorption of $\boldsymbol{t}$

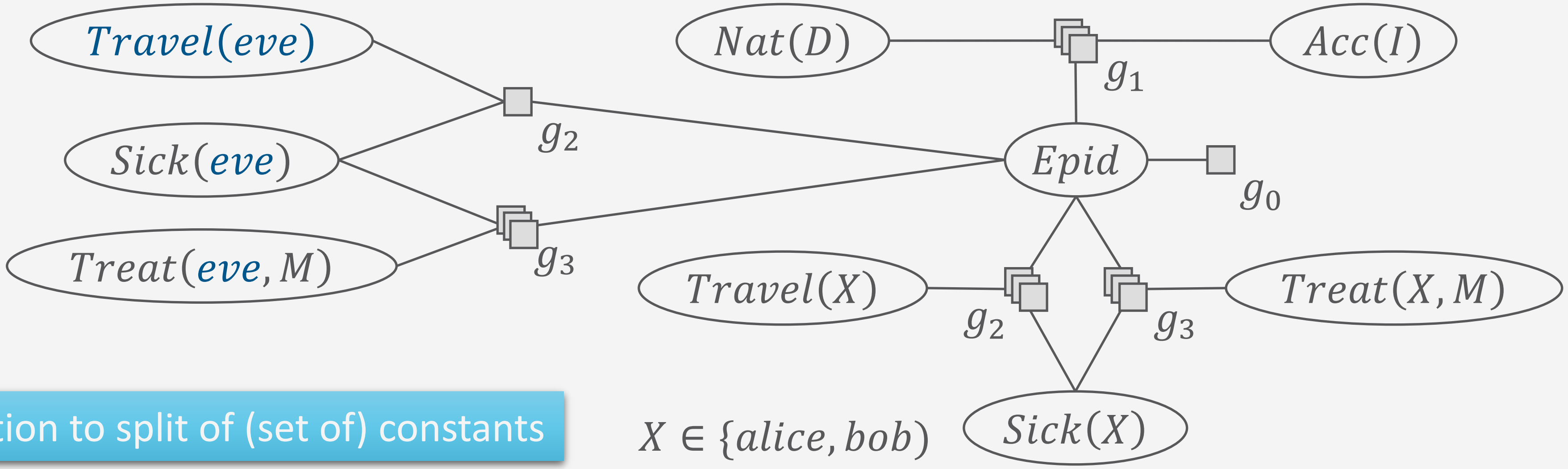
→ Lifting operators of LVE

Lifting operators to enable the main operators above necessary

LVE in Detail

- Example: $P\big(Travel(eve)\big)$ in $G = \{g_i\}_{i=0}^{3}$
  - Pre-processing:
    Split all parfactors whose constraint contains constants occurring in query terms: eve
    - If parameterised query $P\big(\boldsymbol{A}_{|C}\big)$: split parfactors based on C

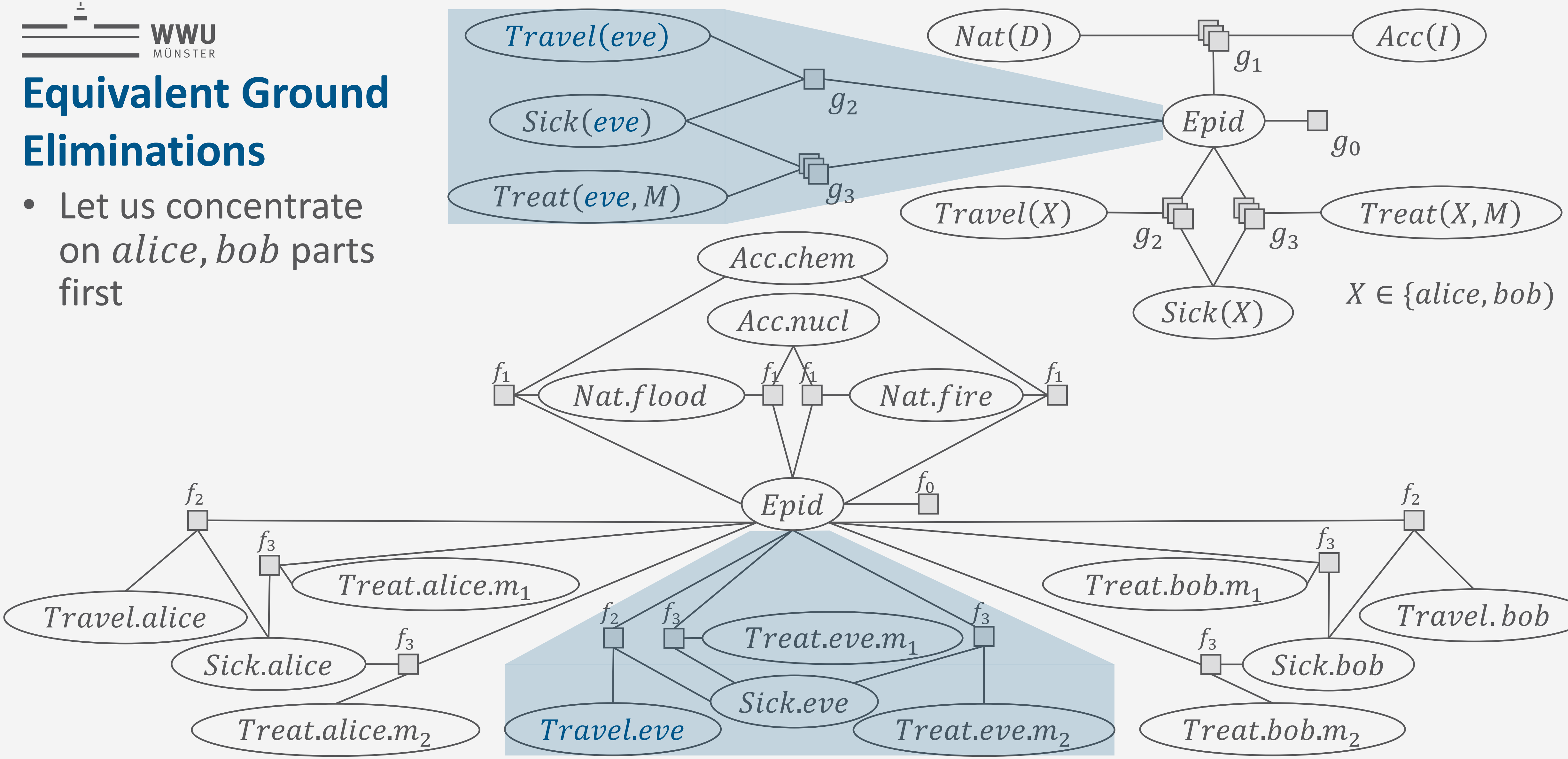
Called shattering



So, we need a formal *split* operation to split of (set of) constants

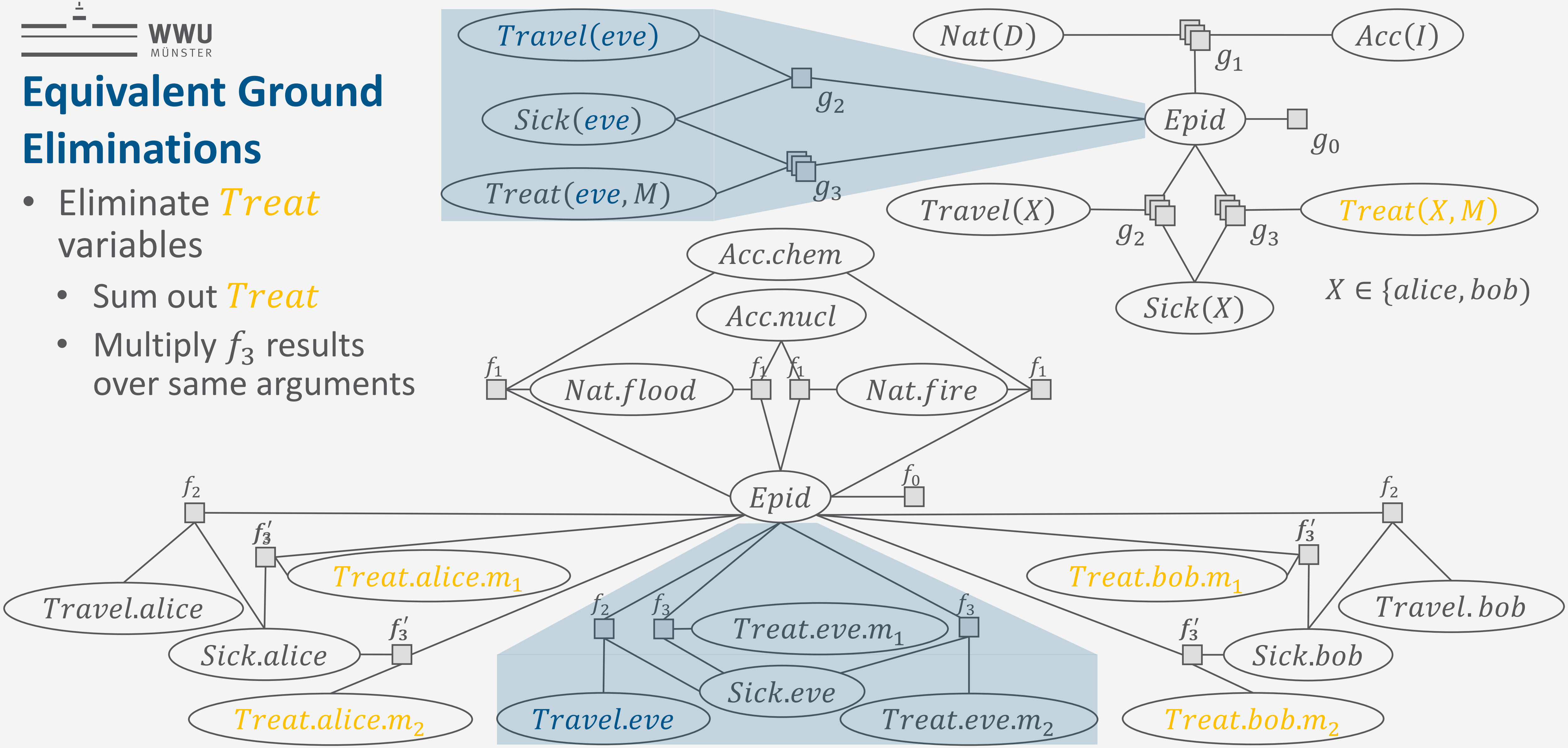
Equivalent Ground Eliminations

- Let us concentrate on $alice, bob$ parts first



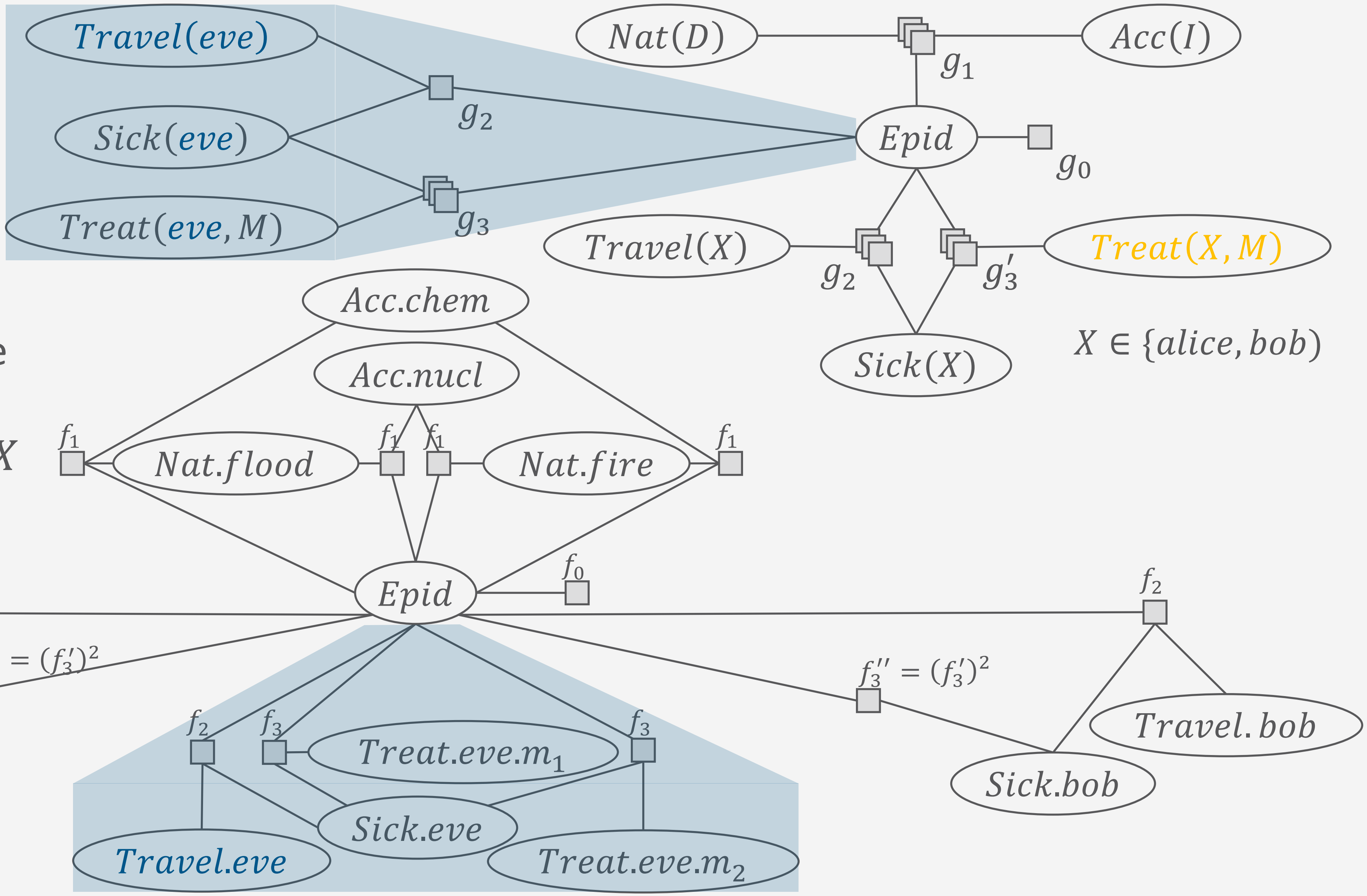
Equivalent Ground Eliminations

- Eliminate *Treat* variables
  - Sum out *Treat*
  - Multiply f_3 results over same arguments



Equivalent Ground Eliminations

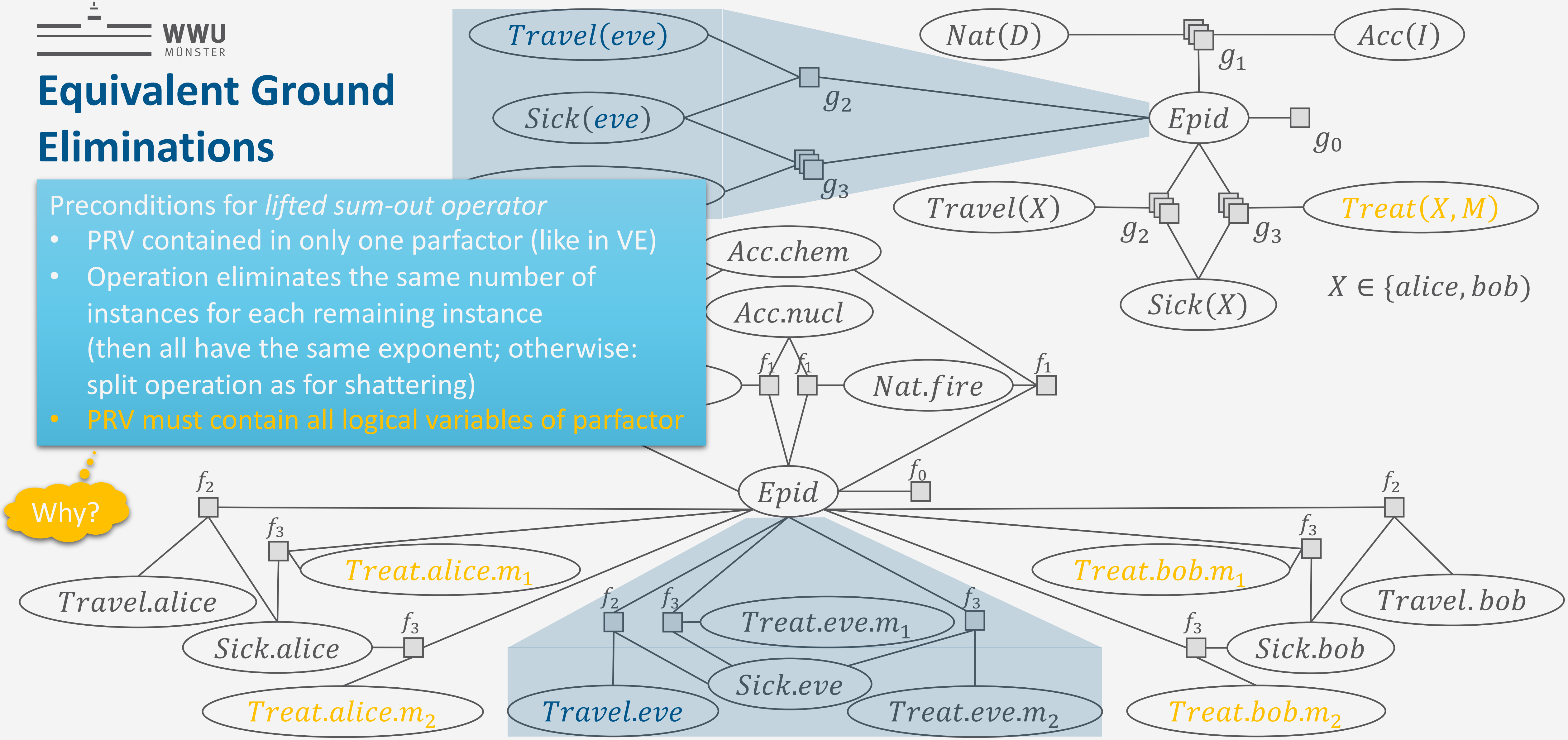
- Eliminate $Treat(X, M)$ lifted
  - Sum out representative
  - Exponentiate result with # of M's for each X



Equivalent Ground Eliminations

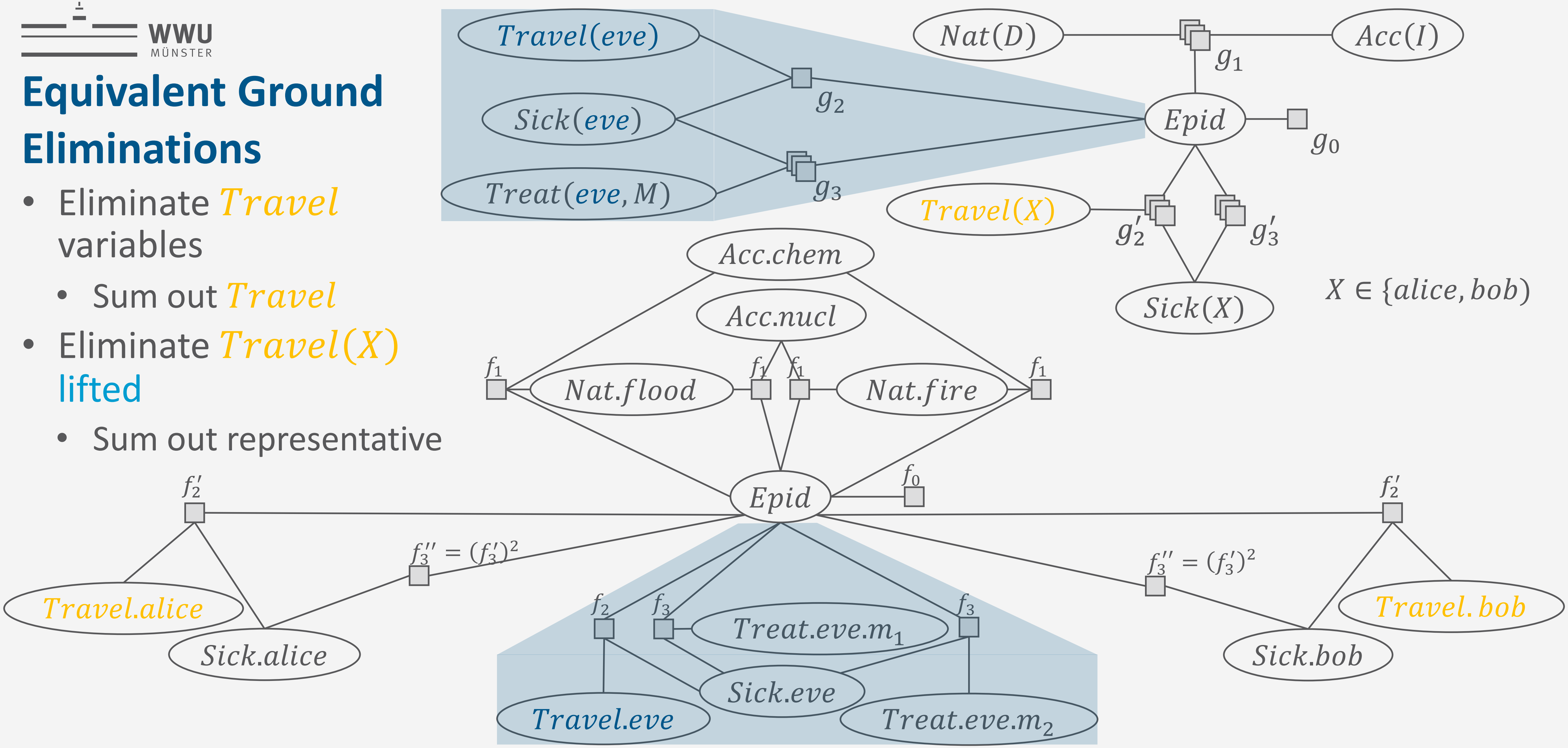
Preconditions for *lifted sum-out operator*
- PRV contained in only one parfactor (like in VE)
- Operation eliminates the same number of instances for each remaining instance (then all have the same exponent; otherwise: split operation as for shattering)
- PRV must contain all logical variables of parfactor

Why?



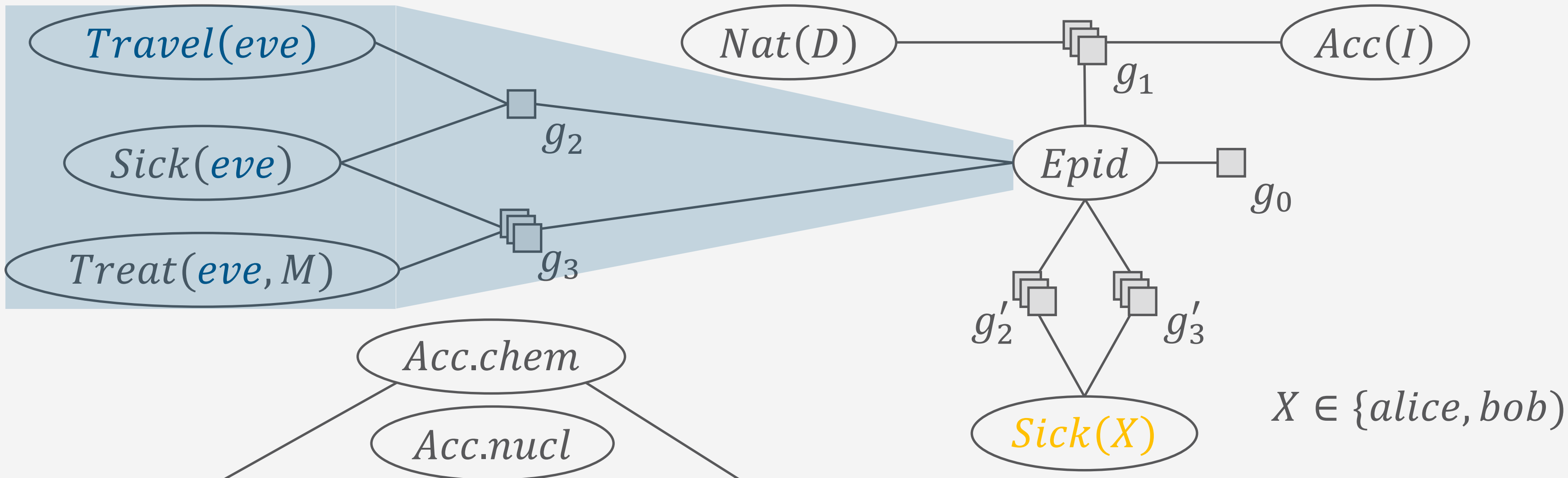
Equivalent Ground Eliminations

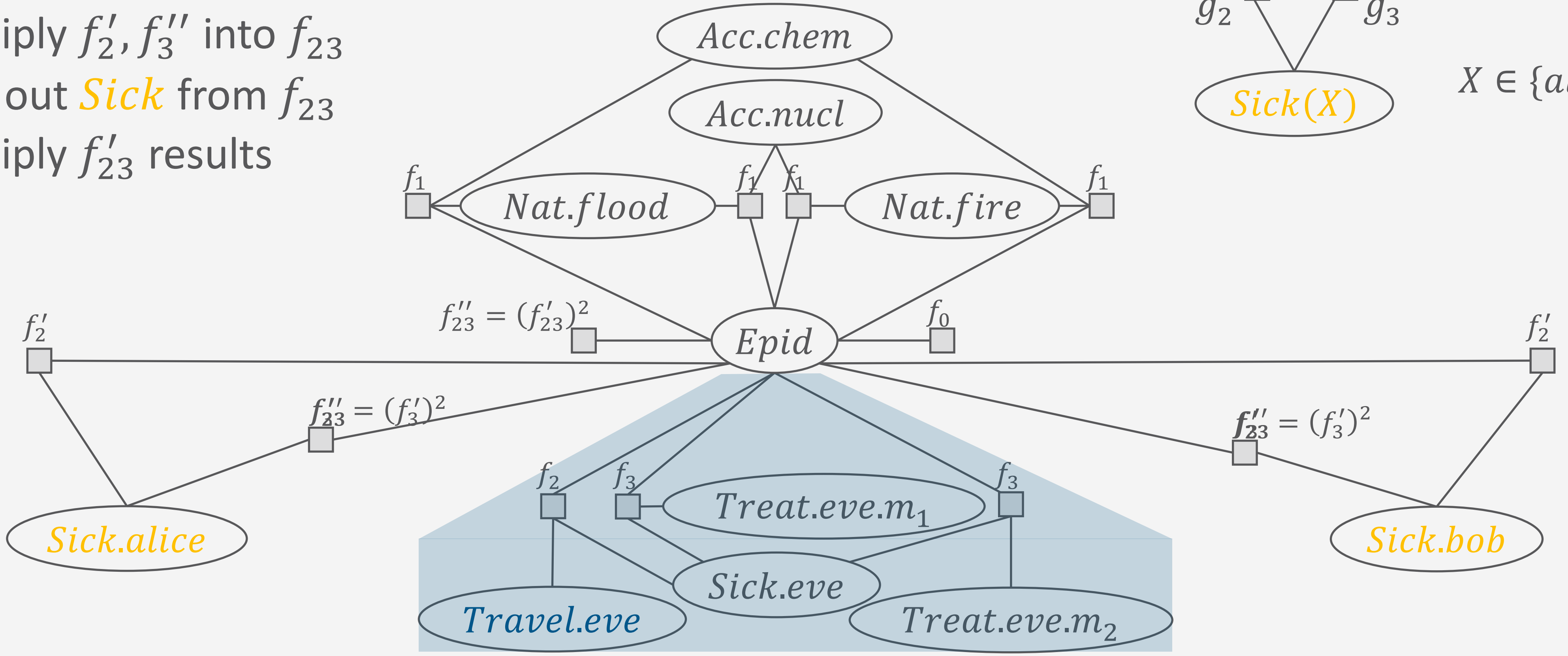
- Eliminate $Travel$ variables
  - Sum out $Travel$
- Eliminate $Travel(X)$ lifted
  - Sum out representative



Equivalent Ground Eliminations

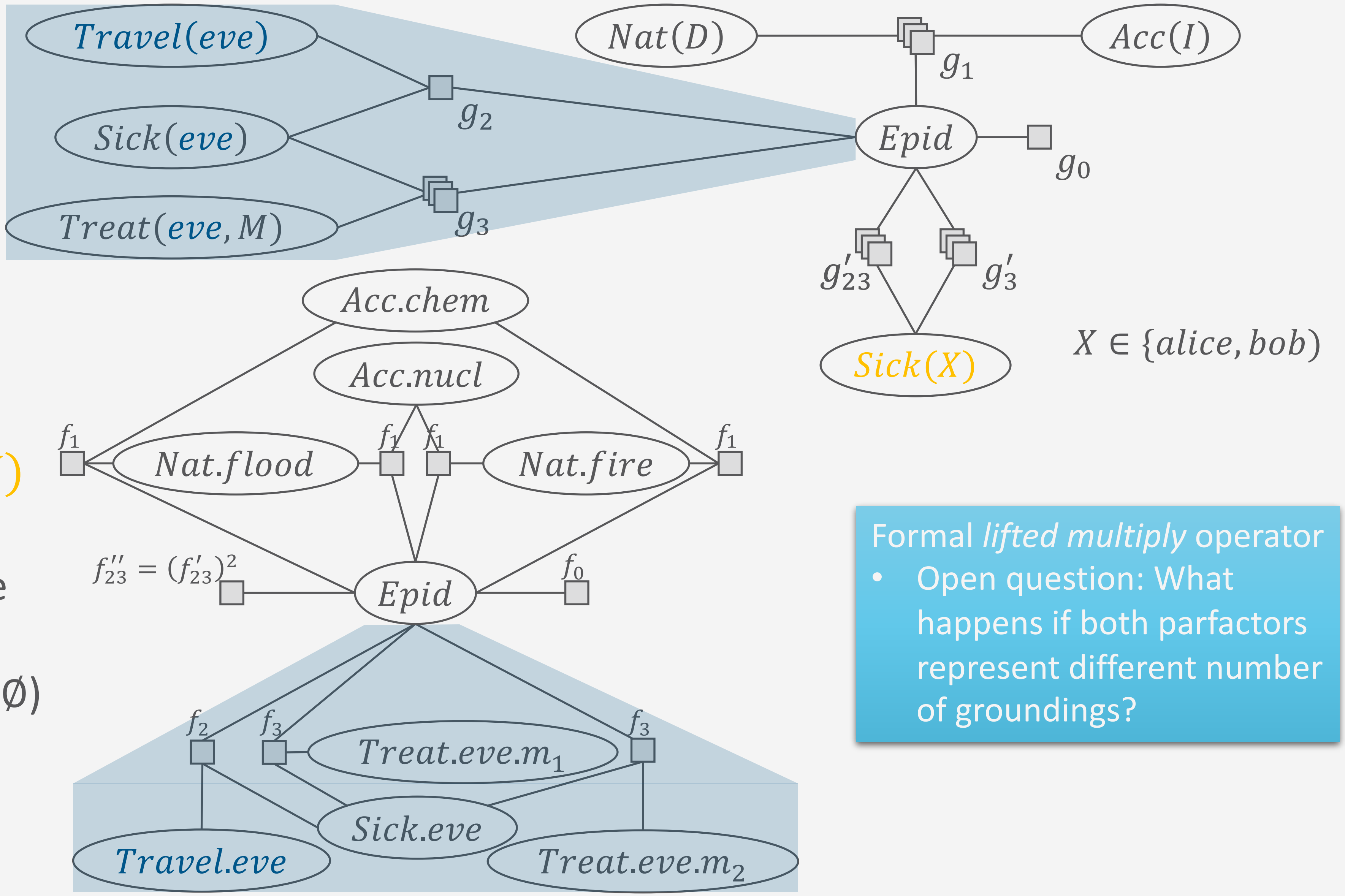
- Eliminate *Sick*
  - Multiply f'_2, f''_3 into f_{23}
  - Sum out *Sick* from f_{23}
  - Multiply f'_{23} results





Equivalent Ground Eliminations

- Eliminate $Sick(X)$ lifted
- First:
  - Multiply g_2', g_3' lifted
- Then, eliminate $Sick(X)$ lifted
  - Sum out representative
  - Exponentiate result with # of X's for $Epid$ ($\emptyset$)

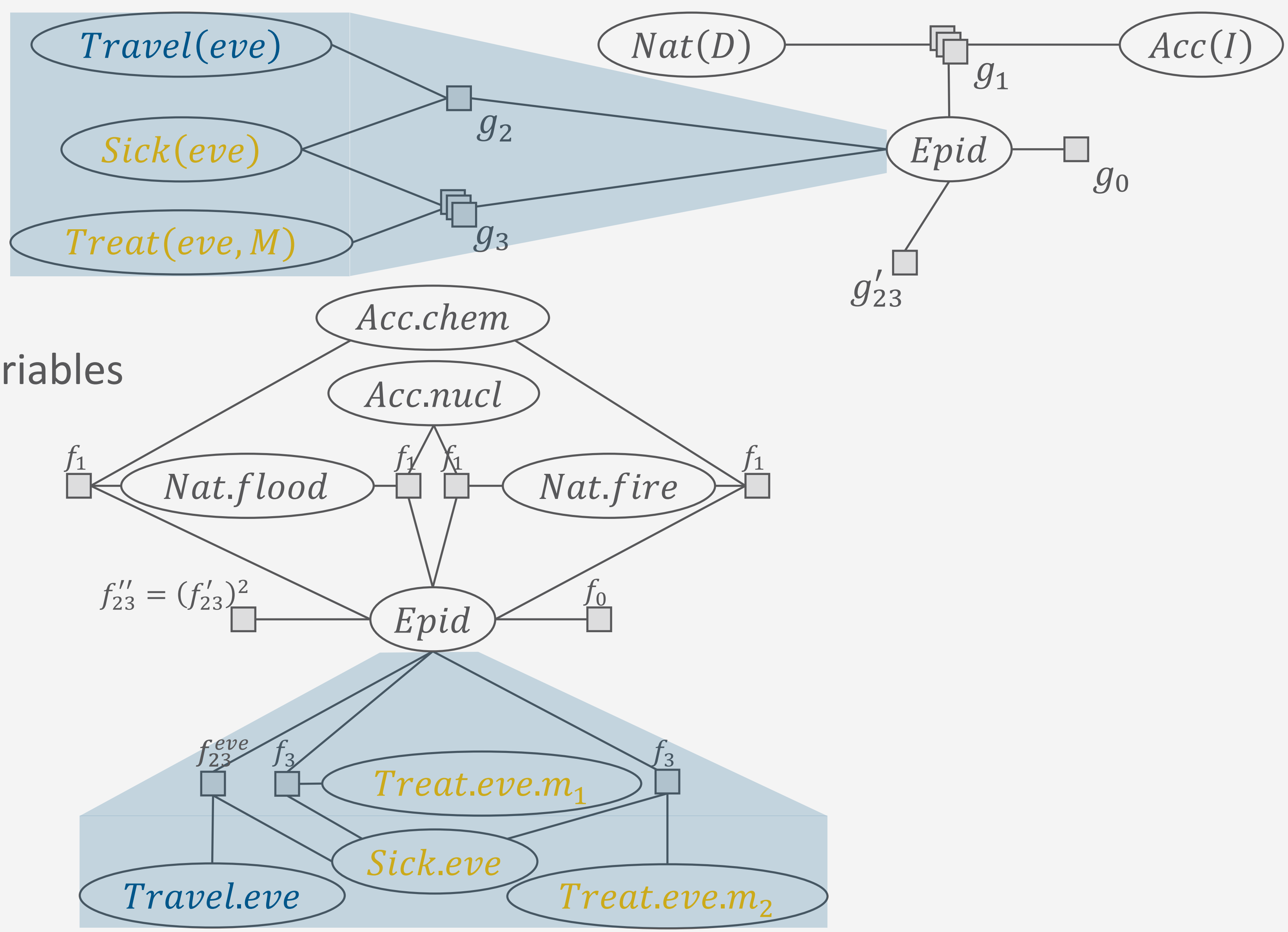


Formal *lifted multiply* operator

- Open question: What happens if both parfactors represent different number of groundings?

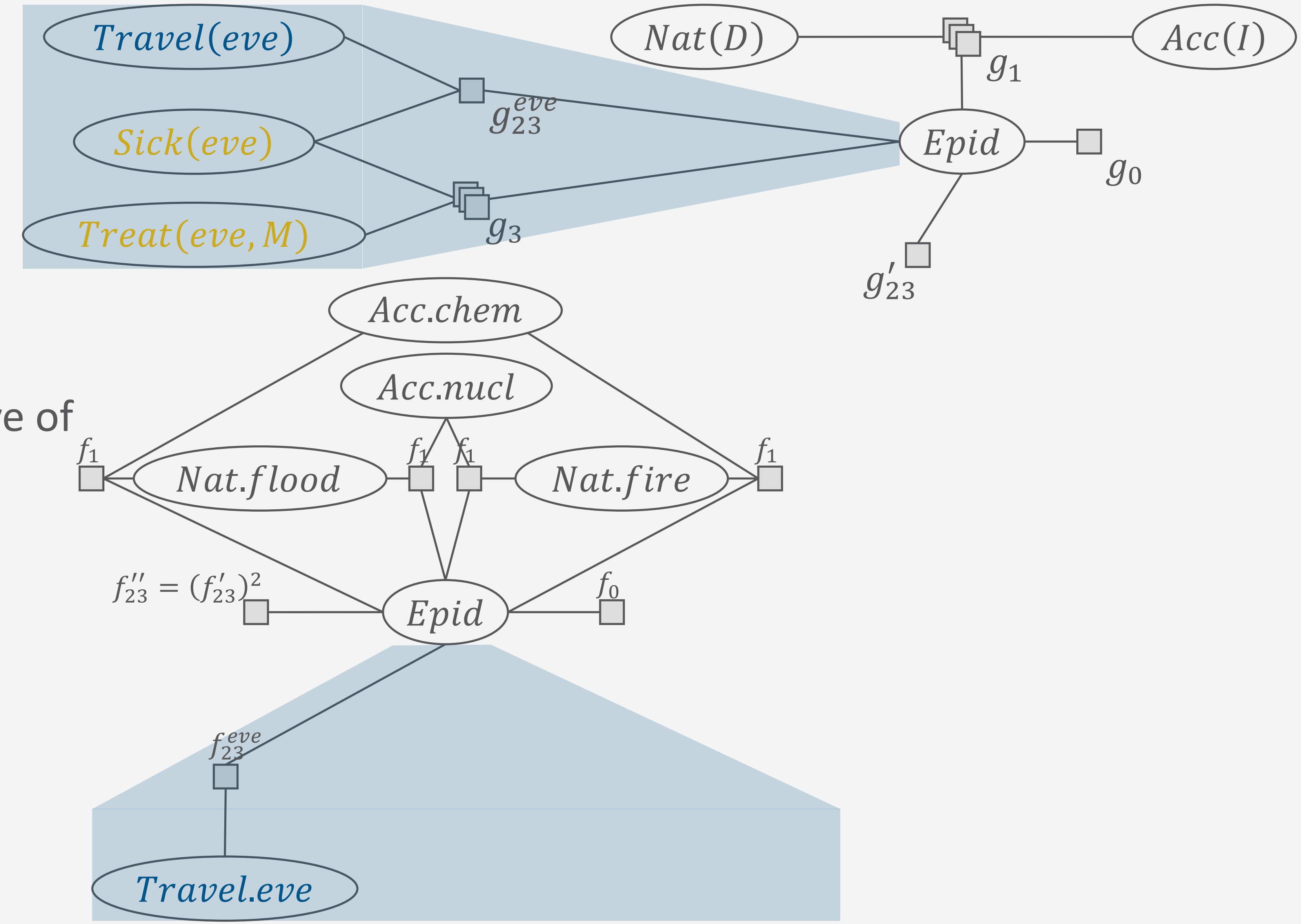
Equivalent Ground Eliminations

- Eliminate $Treat.eve$, $Sick.eve$ variables
  - Sum out $Treat.eve$ variables
  - Multiply results over same arguments
  - Multiply f_2, f_3'' for eve
  - Sum out $Sick.eve$



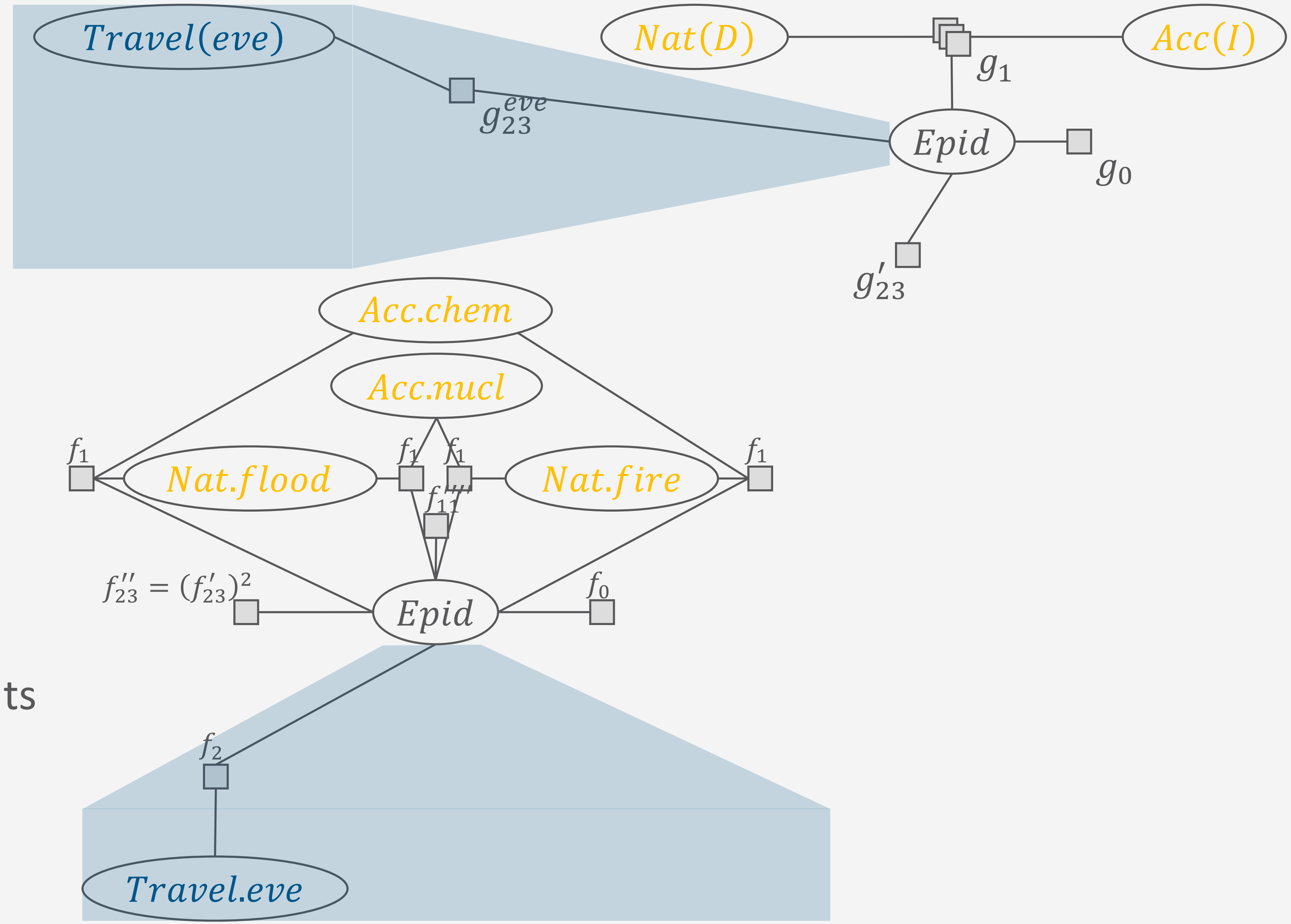
Equivalent Ground Eliminations

- Eliminate $Treat(eve, M)$ lifted, $Sick(eve)$
  - Sum out representative of $Treat(eve, M)$
  - Exponentiate with 2
  - Multiply g_2, g_3'
  - Sum out $Sick(eve)$



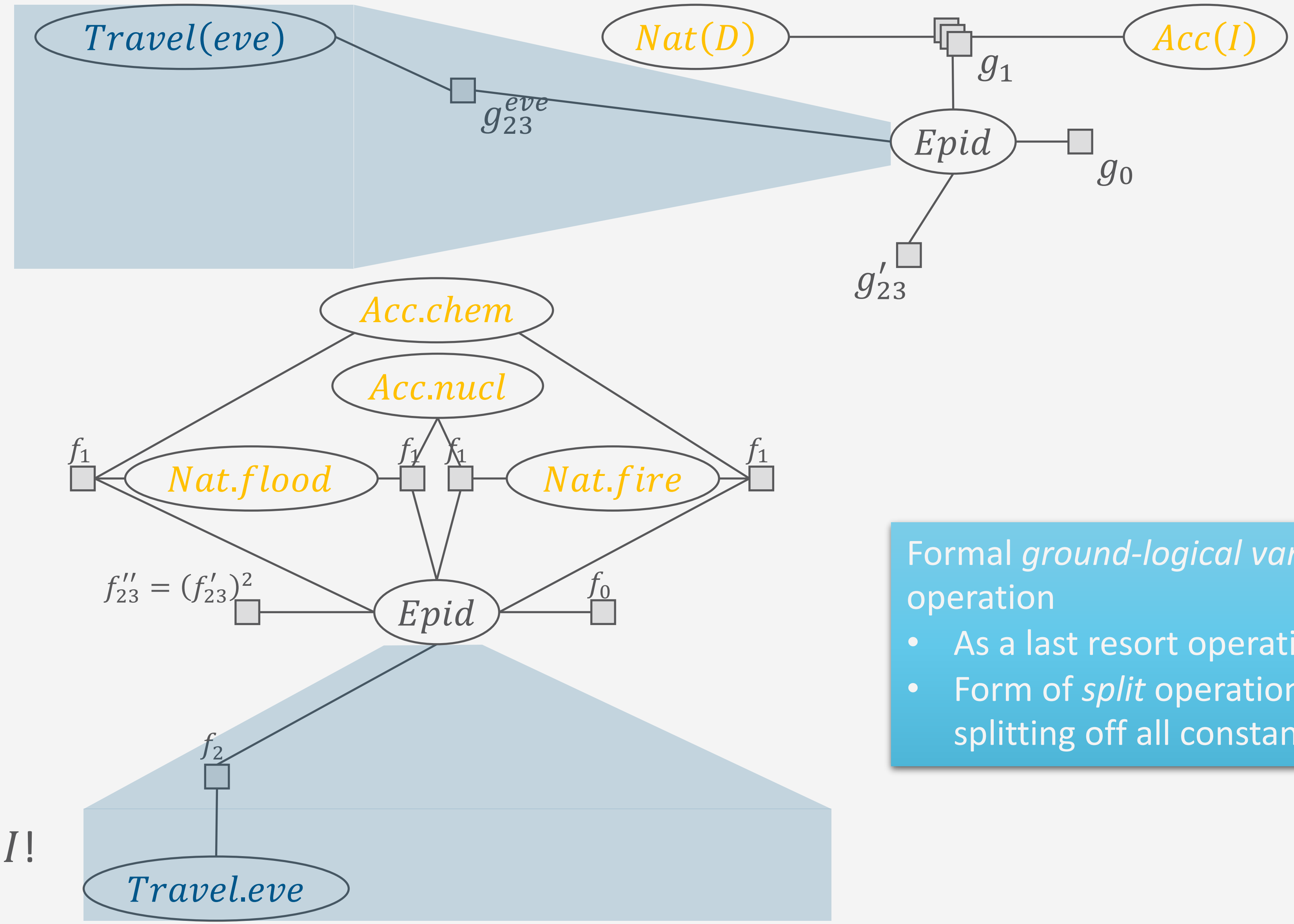
Equivalent Ground Eliminations

- Eliminate Nat, Acc variables
  - Start with $Nat.flood$
    - Multiply f_1, f_1 over $Acc.chem$, $Acc.nucl$
    - Sum out $Nat.flood$
  - Same for $Nat.fire$
    - Identical result
  - Multiply identical results into f''_{11}
  - Sum out $Acc.chem$, $Acc.nucl$



Equivalent Ground Eliminations

- Eliminate $Nat(D)$, $Acc(I)$ lifted
  - Problem:
    Neither contains all logical variables of g_1
  - Solution: Ground I?
    - Eliminate $Nat(D)$
    - Eliminate $Acc(chem)$, $Acc(nucl)$
  - But: local symmetries, encode in histograms
  - Better solution: Count I!

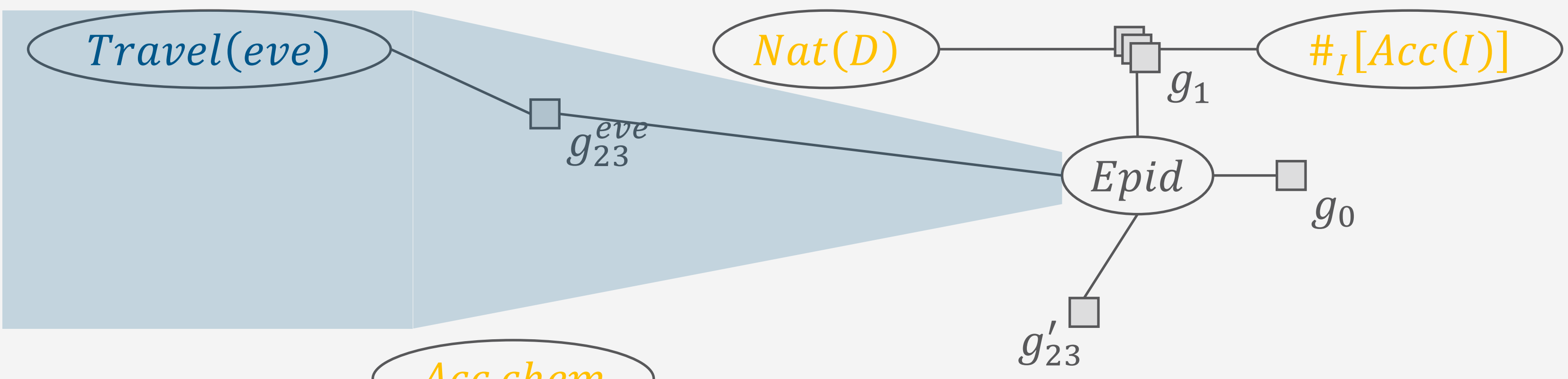


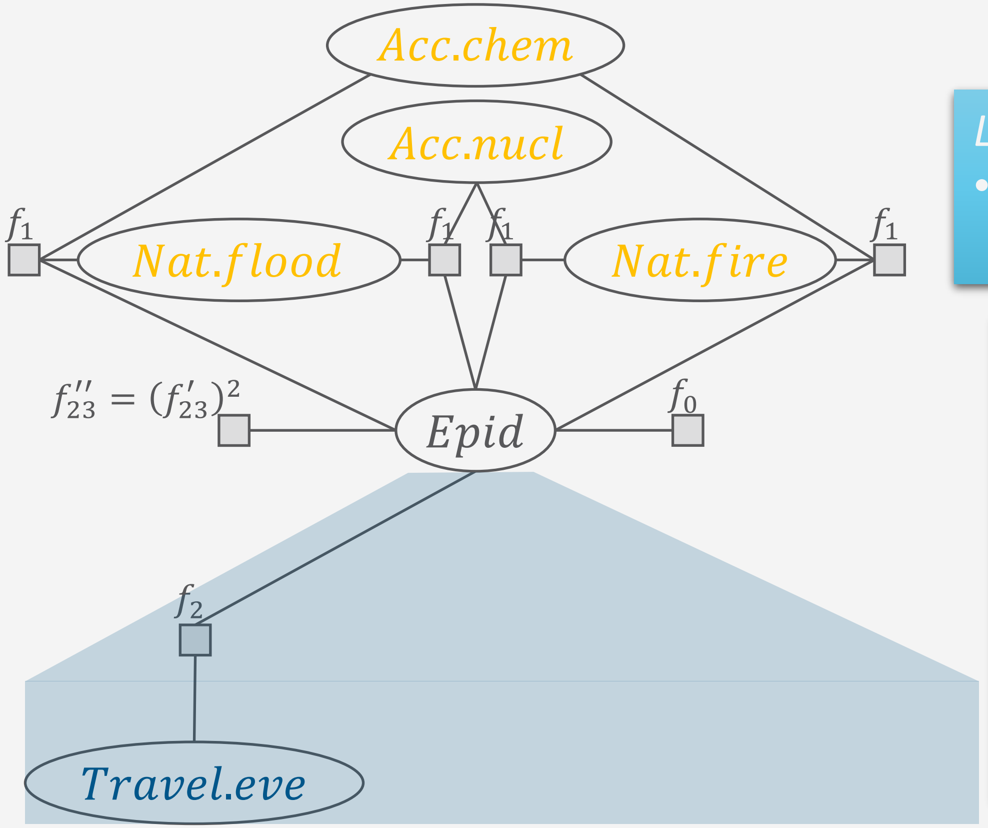
Formal *ground-logical variable* operation

- As a last resort operation
- Form of *split* operation, splitting off all constants

Equivalent Ground Eliminations

- Eliminate $Nat(D)$, $Acc(I)$ lifted
  - Count I in $Acc(I)$
    - CRV $\#_I[Acc(I)]$
  - Eliminate $Nat(D)$ lifted
    - Sum out representative
    - Exponentiate with # of D's
  - Eliminate $\#_I[Acc(I)]$
    - Sum out while considering $Mul(H)$



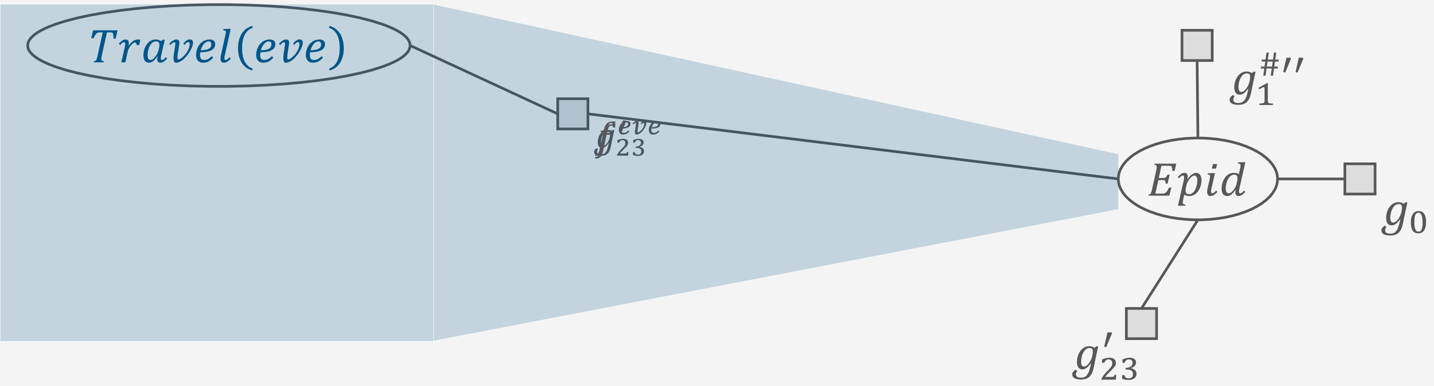


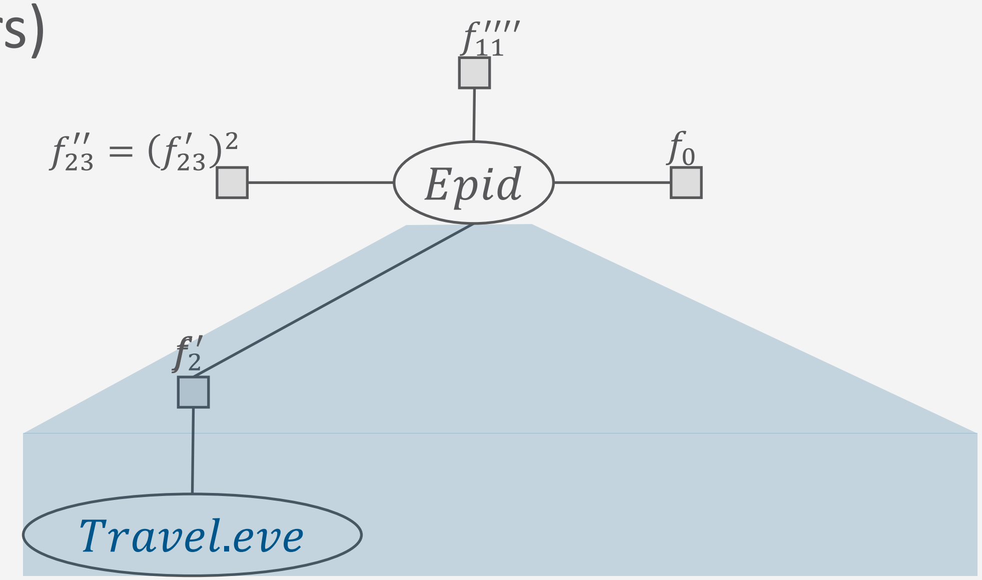
Lifted sum-out operator addendum
- Sum-out operation needs to be able to handle CRVs correctly

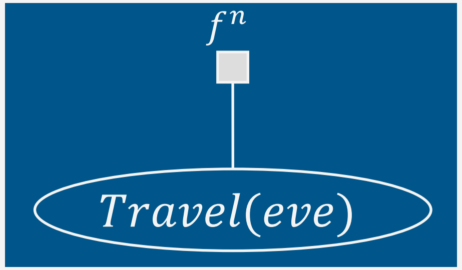
Formal *count conversion* operator
- Number of instances of logical variable to count identical for all instances of other logical variables in parfactor (to have identical histograms)
- Logical variable to count appears in only one PRV

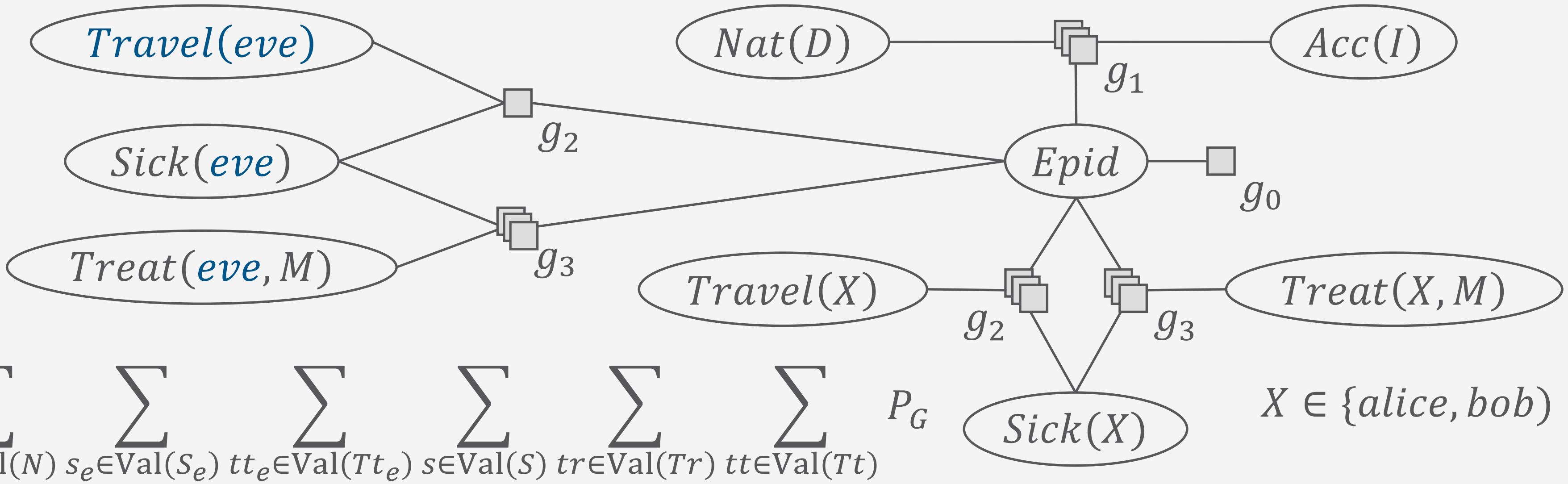
Equivalent Ground Eliminations

- Eliminate *Epid*
  - Identical in both cases
  - Multiply all remaining factors into f
  - Sum out *Epid*
- (Multiply remaining factors)
  - Here only one factor f'
- Normalise result
  $\rightarrow f^n = P(Travel(eve))$









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

$$\propto \sum_{e\in \mathrm{Val}(E)} \sum_{a\in \mathrm{Val}(\#_I[Acc(I)])} \sum_{n\in \mathrm{Val}(N)} \sum_{s_e\in \mathrm{Val}(S_e)} \sum_{tt_e\in \mathrm{Val}(Tt_e)} \sum_{s\in \mathrm{Val}(S)} \sum_{tr\in \mathrm{Val}(Tr)} \sum_{tt\in \mathrm{Val}(Tt)} P_G$$

$$\propto \sum_{e\in \mathrm{Val}(E)} \phi_0(e) \sum_{a\in \mathrm{Val}(\#_I[Acc(I)])} \left(\sum_{n\in \mathrm{Val}(N)} \phi_1(e,n,a)\right)^2$$

$$\sum_{s_e\in \mathrm{Val}(S_e)} \phi_2(Travel, e, s_e) \left(\sum_{tt_e\in \mathrm{Val}(Tt_e)} \phi_3(e, s_e, tt_e)\right)^2$$

$$\sum_{s\in \mathrm{Val}(S)} \sum_{tr\in \mathrm{Val}(Tr)} \phi_2(tr, e, s)$$

$$\left(\sum_{tt\in \mathrm{Val}(Tt)} \phi_3(e, s, tt)\right)^2$$

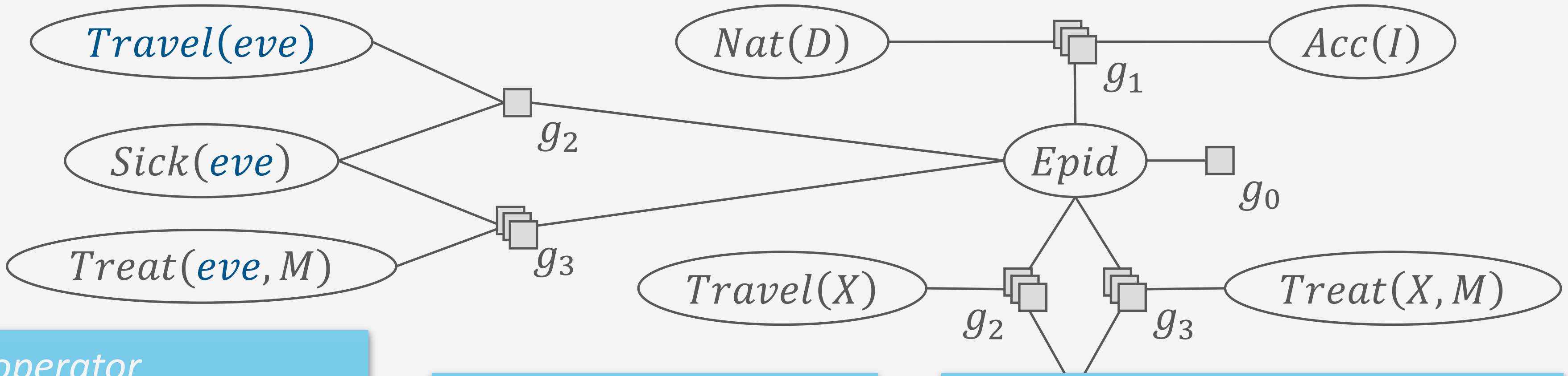
Lifted operators for
- Summing out
- Multiplication
- Absorption of ***t***

→ Lifting operators of LVE

Lifting operators to enable the main operators above necessary

$E \triangleq Epid, N \triangleq Nat(D), A \triangleq Acc(I), S \triangleq Sick(X), Tt \triangleq Treat(X,M), Tr \triangleq Travel(X)$; *subscript e for eve*

Lifted Operators and Their Preconditions



Preconditions for *lifted sum-out operator*

- PRV contained in only one parfactor (like in VE)
- PRV must contain all logical variables of parfactor
- Operation eliminates the same number of instances for each remaining instance (then all have the same exponent; otherwise: split operation as for shattering)

Lifted sum-out operator addendum

- Sum-out operation needs to be able to handle CRVs correctly

So, we need a formal *split* operation to split of (set of) constants

Formal *lifted multiply* operator

- Open question: What happens if both parfactors represent different number of groundings?

Formal *count conversion* operator

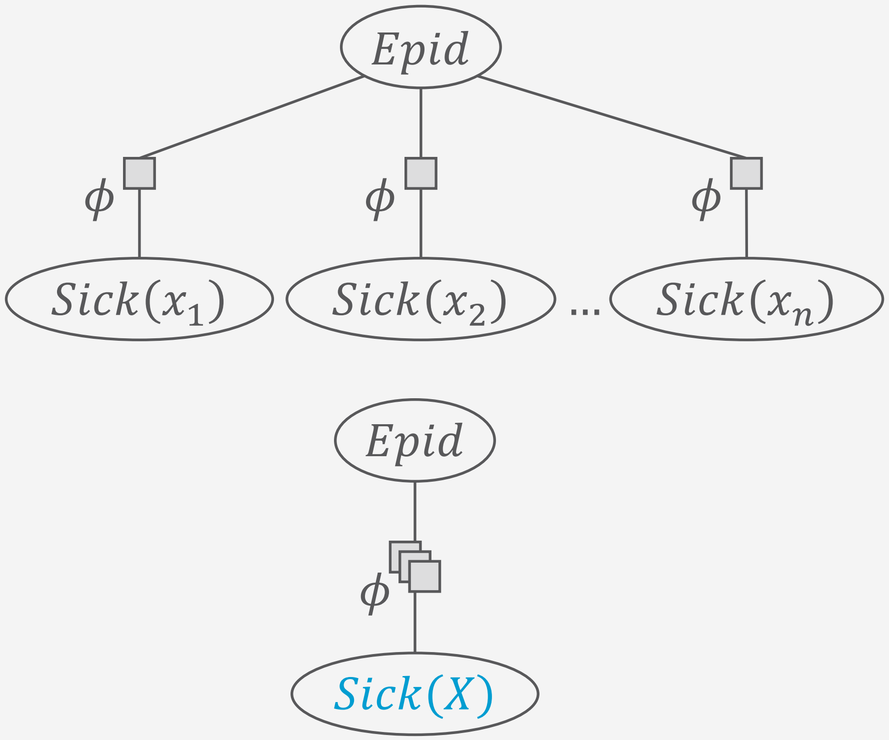
- Number of instances of logical variable to count identical for all instances of other logical variables in parfactor (to have identical histograms)
- Logical variable to count appears in only one PRV

Formal *ground-logical variable* operation

- As a last resort operation
- Form of *split* operation, splitting off all constants

Count Normalisation

- For the different possible groundings of common logical variables $\boldsymbol{X}^{com}$, the same number of groundings of exclusive logical variables $\boldsymbol{X}^{excl}$ exist
  - $\boldsymbol{X}^{excl}$ contains logical variables that are eliminated during a sum-out operation or the logical variable to count
- 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



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 $\mathrm{count}_{Y|Z}(t) : C_\mathcal{X} \to \mathbb{N}$ is defined by
  $$\mathrm{count}_{\boldsymbol{Y}|\boldsymbol{Z}}(t) = \left|\pi_{\boldsymbol{Y}}\left(C_\mathcal{X} \bowtie \pi_{\boldsymbol{Z}}(\{t\})\right)\right|$$

- Count-normalisation:
  $\boldsymbol{Y}$ is count-normalised w.r.t. to $\boldsymbol{Z}$ iff $\exists n \in \mathbb{N}$ s.t.
  $$\forall t \in C_\mathcal{X} : \mathrm{count}_{\boldsymbol{Y}|\boldsymbol{Z}}(t) = n$$
  - Conditional count of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, denoted $\mathrm{ncount}_{\boldsymbol{Y}|\boldsymbol{Z}}\left((\mathcal{X}, C_\mathcal{X})\right)$

Example

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

$\text{count}_{M|X}((a, m_2))$
$\pi_X(\{(a, m_2)\}) = \{(a)\}$
$C_{X,M} \bowtie \{(a)\} = \{(a, m_2)\}$
$\pi_M(\{(a, m_2)\}) = \{(m_2)\}$
$|\{(m_2)\}| = 1$

$\text{count}_{M|X}((e, m_1))$
$\pi_X(\{(e, m_1)\}) = \{(e)\}$
$C_{X,M} \bowtie \{(e)\} = \{(e, m_1), (e, m_2)\}$
$\pi_M(\{(e, m_1), (e, m_2)\}) = \{(m_1), (m_2)\}$
$|\{(m_1), (m_2)\}| = 2$

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

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

Lifted Summing Out

- Summing out transforms the current model G
  - Removes a PRV from $rv(G)$
- Effect on logical variables :
  - Number of logical variables decreases over the whole LVE run for one query
  - Until only propositional random variables and CRVs (counted logical variables are bound) are left → Standard variable elimination
- Preconditions act as a filter on possible $\text{sum}-\text{out}$ operations

Preconditions for *lifted sum-out operator*
- PRV contained in only one parfactor (like in VE)
- PRV must contain all logical variables of parfactor
- Operation eliminates the same number of instances for each remaining instance (then all have the same exponent; otherwise: split operation as for shattering)

Lifted sum-out operator addendum
- Sum-out operation needs to be able to handle CRVs correctly

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:
  1. $\forall B \in rv(G \setminus \{g\}) : gr(B_{|C}) \cap gr(A_{i|(\mathcal{X},C_{\mathcal{X}})}) = \emptyset$
  2. $\forall X \in \{X \mid |\pi_X(C_{\mathcal{X}})| > 1\} : X \in lv(A_i)$
  3. $\boldsymbol{X}^{excl} = lv(A_i) \setminus (\mathcal{X} \setminus lv(A_i))$ count-normalised w.r.t. $\boldsymbol{X}^{com} = lv(A_i) \cap \mathcal{X}$ in C: $r = \text{ncount}_{\boldsymbol{X}^{excl}|\boldsymbol{X}^{com}}(C)$
- Output: $\phi'(\mathcal{A}')_{|C'}$ with $C' = (\pi_{\boldsymbol{X}^{com}}(\mathcal{X}), \pi_{\boldsymbol{X}^{com}}(C_{\mathcal{X}}))$
  - $\mathcal{A}' = (A_1, \dots, A_{i-1}) \circ (A_{i+1}, \dots, A_n)$ (concatenation of two sequences)
  - For each assignment $\boldsymbol{a}' = (\dots, a_{i-1}, a_{i+1}, \dots)$ to $\mathcal{A}'$, i.e., $\forall\, \boldsymbol{a}' \in ran(\mathcal{A}')$

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

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

Multinomial coefficient to eliminate CRVs correctly

$$Mul(a_i) = \begin{cases} \frac{n!}{\prod_{i=1}^{m} n_i!} & a_i = h \\ 1 & oth. \end{cases}$$

Lifted Multiplication

- Operator for multiplication as an ”enabler” for sum−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
  - Since two parfactors represent two (different) sets of grounded factors, lifted multiplication has to work as a representative multiplication for those two sets
    - Easy case:
      1-to-1 correspondence between groundings of those parfactors
    - *But, what happens if the number of represented factors differ?*
      - 1-to-m correspondence
      - n-to-m correspondence

Formal *lifted multiply* operator
- Open question: What happens if both parfactors represent different number of groundings?

Lifted Multiplication: Trivial Case

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

Each grounding of X in $gr\left(\phi_1\big(S(X)\big)\right)$ interacts with 1 corresponding grounding of X in $gr\left(\phi_2\big(S(X), A(X)\big)\right)$

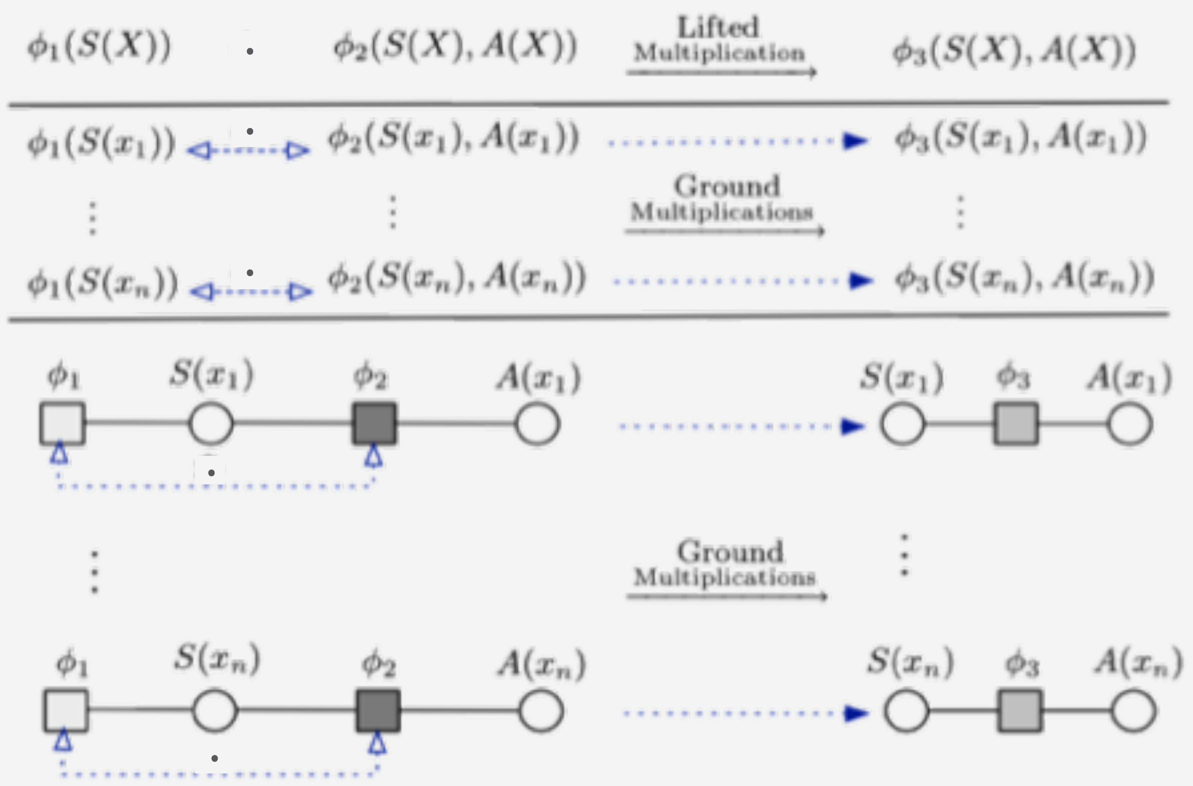


Figure taken from: Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013.

Lifted Multiplication: More General

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

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

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

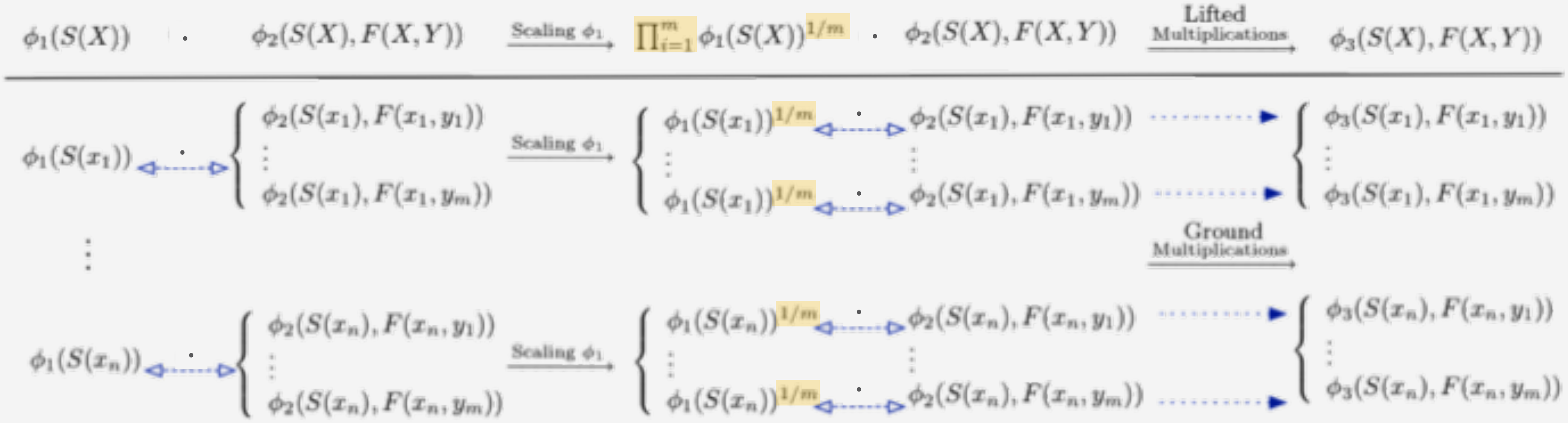


Figure taken from: Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. In: *Journal of Artificial Intelligence Research*, 2013.

Lifted Multiplication: General Case

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

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

- 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 $\theta = \{\boldsymbol{Z}_1 \rightarrow \boldsymbol{Z}_2\}$ between the logical variables 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., $r_i = \text{ncount}_{\boldsymbol{Y}_i|\boldsymbol{Z}_i}(C_i)$ exists
- Output: $\phi(\mathcal{A})_{|C}$ with $C = (\mathcal{X}_1\theta \bowtie \mathcal{X}_2, C_{\mathcal{X}_1\theta} \bowtie C_{\mathcal{X}_2})$
  - $\mathcal{A} = \mathcal{A}_1\theta \bowtie \mathcal{A}_2$
  - 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: $G \sim G \setminus \{g_1, g_2\} \cup \{\text{multiply}(g_1, g_2, \theta)\}$

Lifted Multiplication: Example

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

- T constraints with $|\mathcal{D}(M)| = 2$
- 1-to-m
  - $\boldsymbol{X}_1 = lv(g_1) = \{X\}$
  - $\boldsymbol{X}_2 = lv(g_2) = \{X, M\}$
  - $\boldsymbol{Z}_1 = lv(Sick(X)) = \{X\} = \boldsymbol{Z}_2$
    - No alignment necessary
  - $\boldsymbol{Y}_1 = \boldsymbol{X}_1 \setminus \boldsymbol{Z}_1 = \emptyset$ 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: $r_2 = \text{ncount}_{\boldsymbol{Y}_2|\boldsymbol{Z}_2}(C_2) = 2$

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

$\pi_{Sick(X)}$

$Sick(X)$	ϕ_1
$false$	1
$true$	4

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

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

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

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

Count Conversion

- Counting a logical variable binds a logical variable, i.e., removes logical variable from the logical variables of the parfactor
  - E.g.,
    - $g_1 = \phi_1(Epid, Nat(D), Acc(I)) \rightarrow lv(g_1) = \{D, I\}$
    - $g_1' = \phi_1'(Epid, Nat(D), \#_I[Acc(I)]) \rightarrow lv(g_1') = \{D\}$
  - Helps with Precondition 2 of summing out!
    - Precondition 2: PRV to sum out has to contain all logical variables of parfactor
- Operator count-convert
  - Count a logical variable → convert a PRV into a (P)CRV
  - Works as an "enabler" for sum-out operator
  - Preconditions for count-convert as well

Formal *count conversion* operator
- Number of instances of logical variable to count identical for all instances of other logical variables in parfactor (to have identical histograms)
- Logical variable to count appears in only one PRV

Count Conversion: Operator

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

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

    - With $h(a_i)$ denoting the count of a_i in histogram h
- Postcondition: $G \sim G \setminus \{g\} \cup \{\text{count-convert}(g, X)\}$

No inequality constraint between X and any other counted logical variable $X^{\#}$

Count Conversion: Example

- From $\phi_1(Epid, Nat(D), Acc(I))$
- To $\phi_1'(Epid, Nat(D), \#_I[Acc(I)])$
- Preconditions fulfilled
  - I occurs only in $Acc(I)$
  - I is count-normalised w.r.t. D in $((D, I), dom(D) \times dom(I))$
  - No other counted logical variable
- Converting $Acc(I)$ into $\#_I[Acc(I)]$

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

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

$Epid$	$Nat(D)$	$Acc(I)$	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

Generalised Counting

- Count conversion as discussed here, first introduced by Milch et al. (2008)
- Generalised counting by Nima Taghipour et al. (2013)
  1. Count logical variables 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 logical variables 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

So, we need a formal *split* operation to split of (set of) constants

- Need splitting for
  - Shattering of query terms and evidence
  - Precondition 1 of $\text{sum}-\text{out}$ 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 logical variable are either *identical* or *disjoint*
  - I.e., no overlaps between sets of constants per logical variable
  - 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 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

$$C_1/_{\boldsymbol{Y}}C_2 = \left\{ \begin{array}{ll} \left((\mathcal{X}_1), \left\{ t \in C_{\mathcal{X}_1} \mid \pi_{\boldsymbol{Y}}(\{t\}) \in \pi_{\boldsymbol{Y}}\left(C_{\mathcal{X}_2}\right) \right\} \right) & \text{Part shared with } C_2 \\ \left((\mathcal{X}_1), \left\{ t \in C_{\mathcal{X}_1} \mid \pi_{\boldsymbol{Y}}(\{t\}) \notin \pi_{\boldsymbol{Y}}\left(C_{\mathcal{X}_2}\right) \right\} \right) & \text{Remaining part} \end{array} \right\}$$

- 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

$$C_1/_{\boldsymbol{Y}} C_2 = \left\{ \begin{array}{l} \left(\left((X,Y), \left\{t \in C_{(X,Y)} \mid \pi_X(\{t\}) \in \{x_1, x_2, x_3\}\right\}\right)\right) \\ \left(\left((X,Y), \left\{t \in C_{(X,Y)} \mid \pi_X(\{t\}) \notin \{x_1, x_2, x_3\}\right\}\right)\right) \end{array} \right\} = \left\{ \begin{array}{c} \big((X,Y), \{x_1, x_2, x_3\} \times \{y_1, y_2\}\big) \\ \big((X,Y), \{x_4, x_5\} \times \{y_1, y_2\}\big) \end{array} \right\}$$

- 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 $\#_Y\left[R(\boldsymbol{Y})_{|C'}\right]$
- Precondition: *none*
- Output:

$$\mathrm{partition}(g, \mathbb{C}), \mathbb{C} = C/_Y C' \setminus \emptyset$$

- Postcondition:

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

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:

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

Other Operators

- Further "enablers" of lifted summing out all are *variants of splitting on overlap* and *partitioning*
- Splitting of CRVs: Operator called expand
  - 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 $\{x_1, x_2\}$
- Count-normalisation: Operator called $\mathrm{count{-}normalise}$
  - 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
- Grounding – the last resort: Operator ground as expected
  - Splitting on individual constants
- 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
  - Within each group:
    instances are indistinguishable again

→ Absorb evidence for each group at once using the parfactors

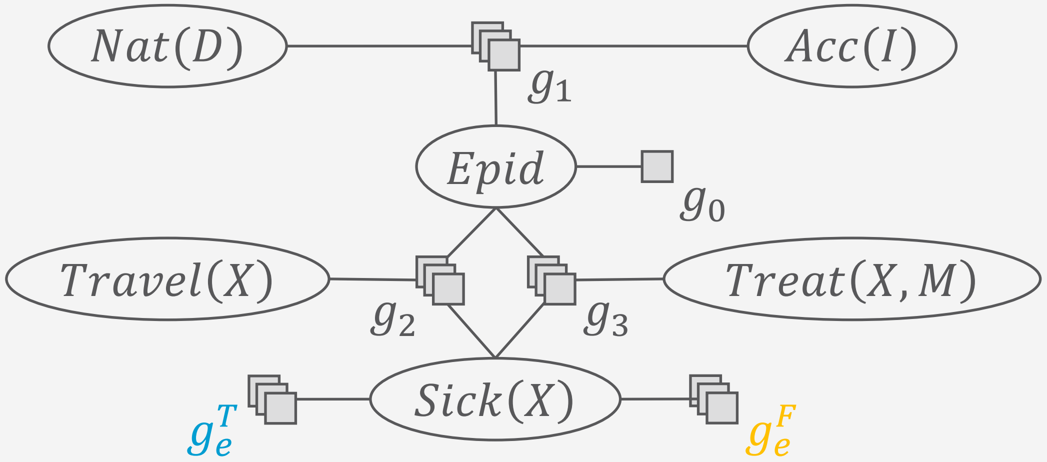
Observations for $Sick(X)$

$Sick(x_1) = Sick(x_2) = \cdots = Sick(x_{10}) = true$

$Sick(x_{11}) = Sick(x_{12}) = \cdots = Sick(x_{20}) = false$

$Sick(X)$	ϕ_e^T
$false$	0
$true$	1

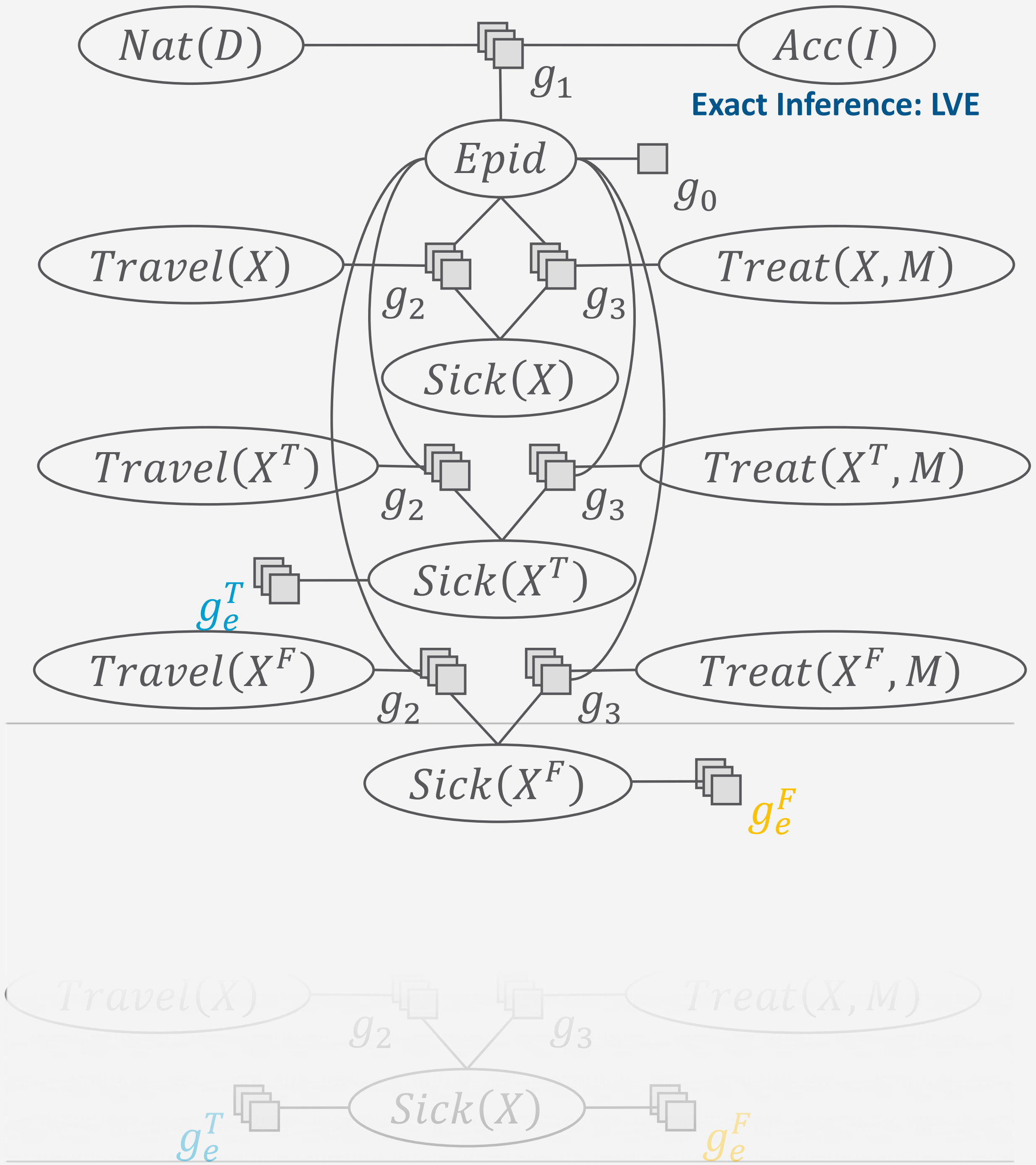
$Sick(X)$	ϕ_e^F
$false$	1
$true$	0



Lifted Absorption: Shattering 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
  - Only then: absorb applicable evidence in each parfactor individually
  - E.g., given evidence parfactors g_e^T, g_e^F, every parfactor containing $Sick(X)$ has to be split on the constraints: g_2, g_3
- After shattering, absorb each evidence parfactor g_e in each applicable parfactor g_i
  - Possible to interleave shattering and absorption

Exact Inference: LVE



Exact Inference: LVE

Lifted Absorption: Example

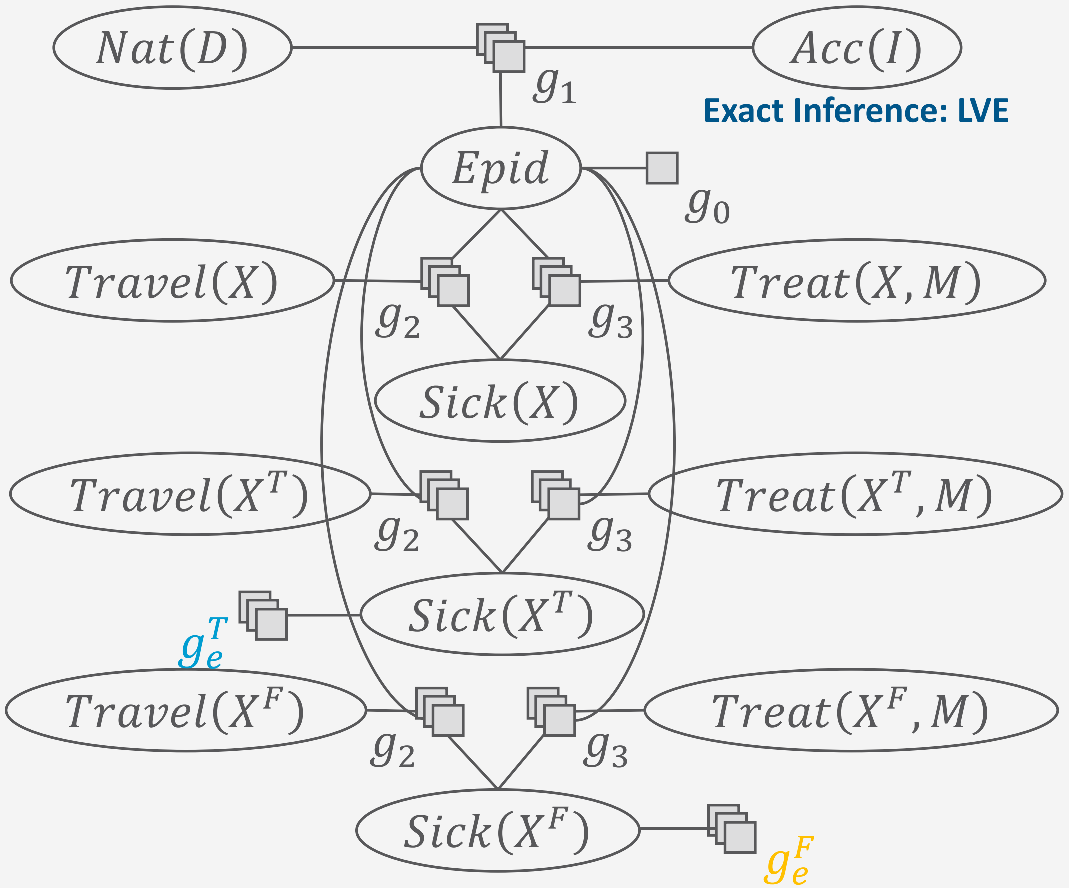
- Absorb g_e^T in g_2:

$Travel(X)$	$Epid$	$Sick(X)$	ϕ_2
~~$false$~~	~~$false$~~	~~$false$~~	~~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

$Sick(X)$	ϕ_e^T
$false$	0
$true$	1

$Travel(X)$	$Epid$	ϕ_2^T
$false$	$false$	0
$false$	$true$	6
$true$	$false$	6
$true$	$true$	9

- Same for g_e^T in g_3, g_e^F in g_2, g_e^F in g_3



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 to A_i), $\boldsymbol{X}^{nce} = lv(A_i) \setminus lv(\boldsymbol{A} \setminus \{A_i\})$ (not-counted exclusive to A_i)
  - $\boldsymbol{X}^{rem} = lv(\mathcal{X}) \setminus \boldsymbol{X}^{excl}$ (remaining in g), $\boldsymbol{X}^{ncr} = lv(\mathcal{A}) \setminus \boldsymbol{X}^{excl}$ (not-counted remaining in g)
- Preconditions:
  1. $gr\big(A_{i|C}\big) \subseteq gr\big(A_{i|C_e}\big)$
  2. $\boldsymbol{X}^{nce}$ 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: $g' = \phi'(\mathcal{A}')_{|C'}, C' = \big(\pi_{\boldsymbol{X}^{rem}}(\mathcal{X}), \pi_{\boldsymbol{X}^{rem}}(C_{\mathcal{X}})\big)$
  - $\mathcal{A}' = (A_1, \dots, A_{i-1}) \circ (A_{i+1}, \dots, A_n)$

$$\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, e, a_{i+1}, \dots)^r$$

    - with $e = o$ if $A_i = R(\boldsymbol{Y})$ 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: 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 $e(o) = \text{ncount}_{X|lv(\mathcal{A})}(C)$ 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 in evidence parfactor
    - Peak-shaped histogram with $\text{ncount}_{X|lv(\mathcal{A})}(C)$ at position o and 0 otherwise
  - E.g., $Nat(D) = false$ for all $gr\big(Nat(D)\big) \rightarrow o = false$ in g_e
    - Given parfactor: $g = \phi(Epid, \#_D[Nat(D)])$
      - $\text{ncount}_{D|\emptyset}(\top) = 2$
      - Forms histogram: [0,2]
    - Output: $g' = \phi'(Epid)$

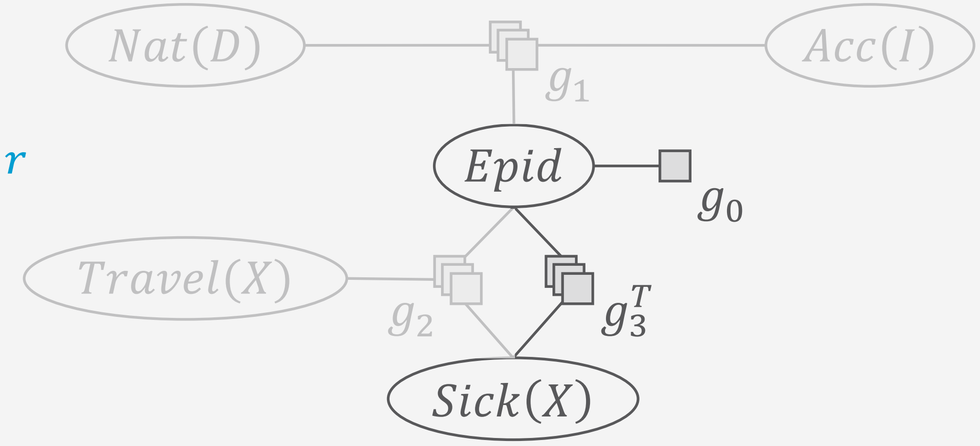
$Nat(D)$	ϕ_e
$false$	1
$true$	0

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

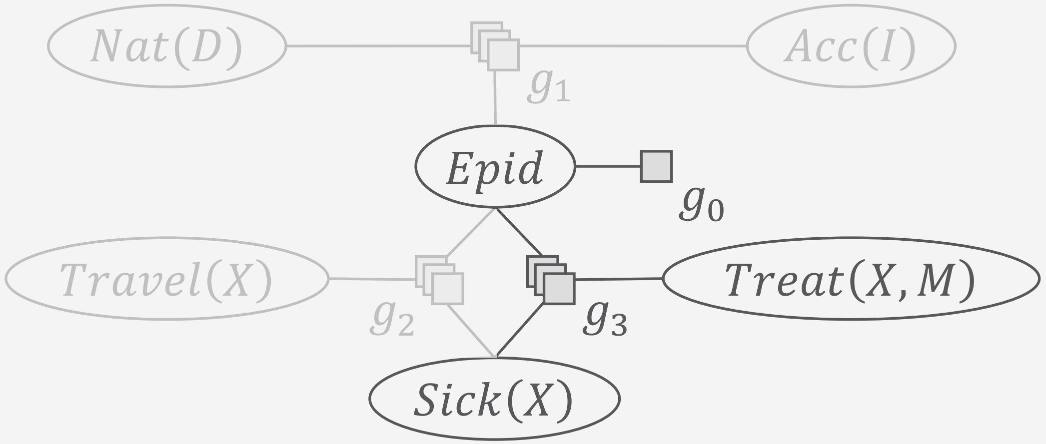
$Epid$	ϕ_e
$false$	1
$true$	4

Lifted Absorption: Eliminating a Logical variable

- 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., $Treat(X, M) = true \; \forall (x, m) \in \top$
  - Parfactor g_3 contains $Treat(X, M)$
  - Output: $\phi'\big(Epid, Sick(X)\big)$
    - Absorbing $Treat(X, M) = true$ eliminates M
    - $r = \mathrm{ncount}_{M|X}(C) = 2$
    - Potentials in selected lines have to be raised to the power of 2

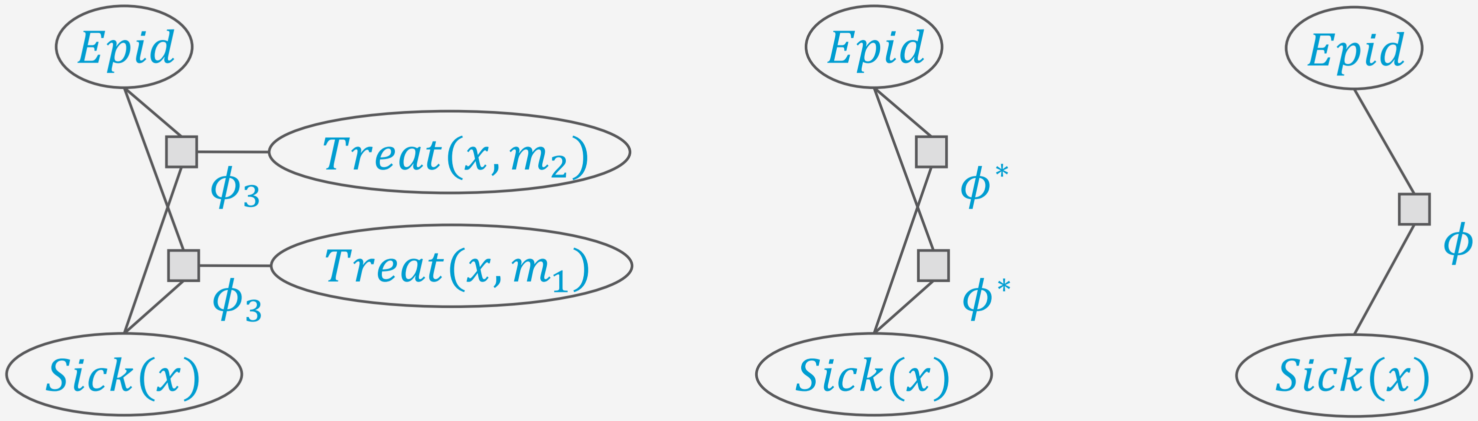


After lifted absorb



Lifted Absorption: Eliminating a Logical variable

- 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 $G = \{g_i\}_{i=1}^{n}$

Shattering

- For each $g_e = \phi_e(A_e)_{|C_e}$:
  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $A_e \in rv(g_i)$:
      - Split g_i on C_e, i.e.,

$$G \leftarrow G \setminus \{g_i\} \cup \text{split}\left(g_i, A_e, A_{e|C_e}\right)$$

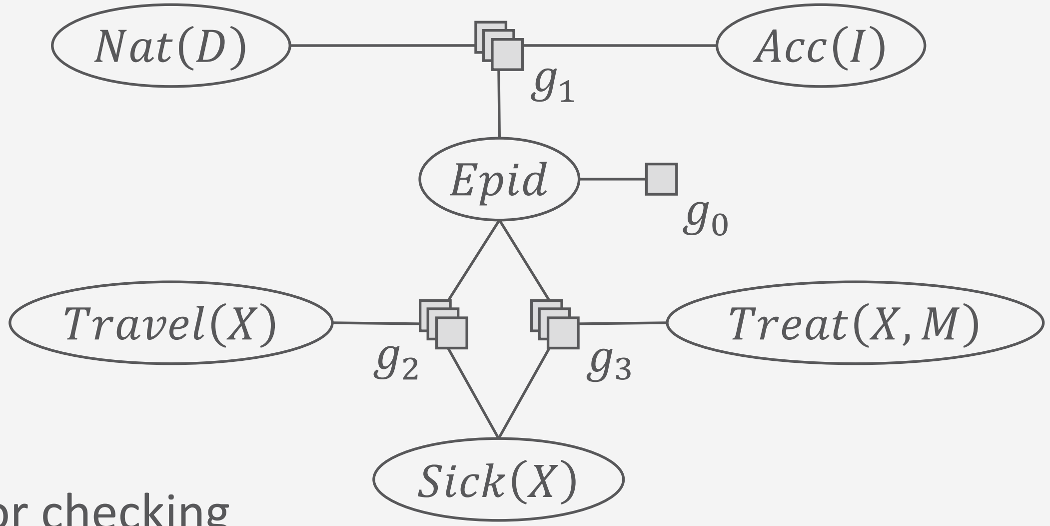
Absorption

- For each $g_e = \phi_e(A_e)_{|C_e}$:
  - For each $g_i = \phi_i(\mathcal{A})_{|C_i}$:
    - If $A_e \in rv(g_i)$:
      - Absorb g_e in g_i

$$G \leftarrow G \setminus \{g_i\} \cup \text{absorb}(g_i, A_e, g_e)$$

Types of Shattering

- So far considered: 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 logical variables occurring in constraints are either identical or disjoint
      - Allows for introducing one logical variable for each set of constants and ⊤ constraints except when an inequality is encoded
      - Avoids splitting during LVE and makes PRV comparisons easier
- On-demand shattering
  - Splitting on constraints only if the application of an LVE operator requires it
  - In initial example calculation for $P\big(Travel(eve)\big)$: Eliminate $Treat(X, M)$ before splitting of $Sick(eve)$
- Does not change complexity of the problem
  - May be hard to determine when to shatter + extra work for checking



LVE: Algorithm

- Assumption:
  - Pre-emptive shattering
  - Ground query terms
    - Set of propositional random variables, instances (groundings) of PRVs,
- Inputs:
  - Model $G = \{g_i\}_{i=1}^{n}$
  - Query terms $\boldsymbol{Q}$
  - Evidence $\boldsymbol{e}$ encoded in evidence parfactors $\{g_e\}_{e=1}^{m}$
- Output:
  - Parfactor $g = \phi(\boldsymbol{Q})$
    - Encodes the a-posteriori probability distribution of $\boldsymbol{Q}$ given $\boldsymbol{e}$: $P(\boldsymbol{Q}|\boldsymbol{e})$

LVE: Algorithm

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

- $G \leftarrow$ Shatter G on $\boldsymbol{Q}$, $\{g_e\}_{e=1}^m$, 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 sum−out **then**
    - $G \leftarrow$ Apply sum−out to A in G
  - **else**
    - $G \leftarrow$ Apply an enabling operator (multiply, count−convert, expand, 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

How to choose?

G may contain several parfactors $\phi_i(\boldsymbol{Q})$

LVE: Heuristics

- Important for an implementation
  - Cannot search all possible permutations of all possible operator applications
- Preconditions of lifted operators already restrict possible elimination order
- One possible greedy heuristics (as used in the upcoming implementation):
  - 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 result of a count conversion is usually smaller in size than the result of grounding the same logical variable

LVE: Implementation

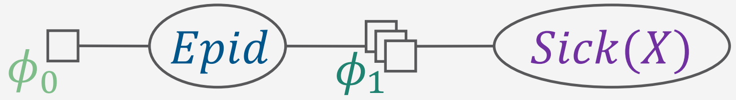
- Available at:
  - https://dtai.cs.kuleuven.be/software/gcfove
  - Includes a VE implementation for comparison
- Input: BLOG files
  - Based on Bayesian Logic Programming Language
    - https://bayesianlogic.github.io
- Differences
  - Constraint language and domains:
    - Intensional language: all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

BLOG Input

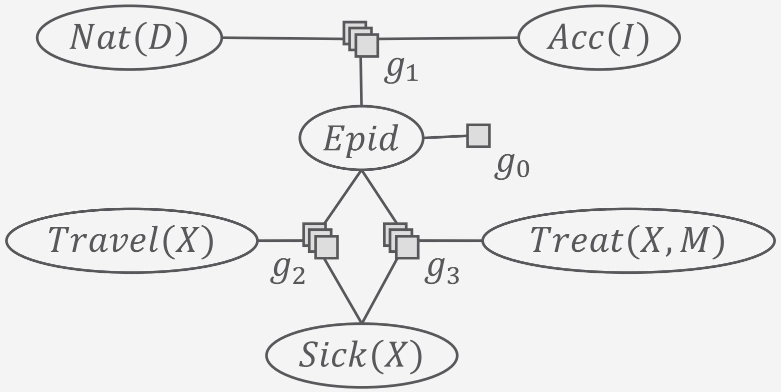
- Components
  - Logical variables
  - Domain definitions
  - Ground random variables
  - PRVs
  - Factors
  - Parfactors
- Potential lists
  - Start at all $true$
  - End at all $false$
    - If you think of the assignments as binary numbers, then the numbers are decreasing

BLOG file

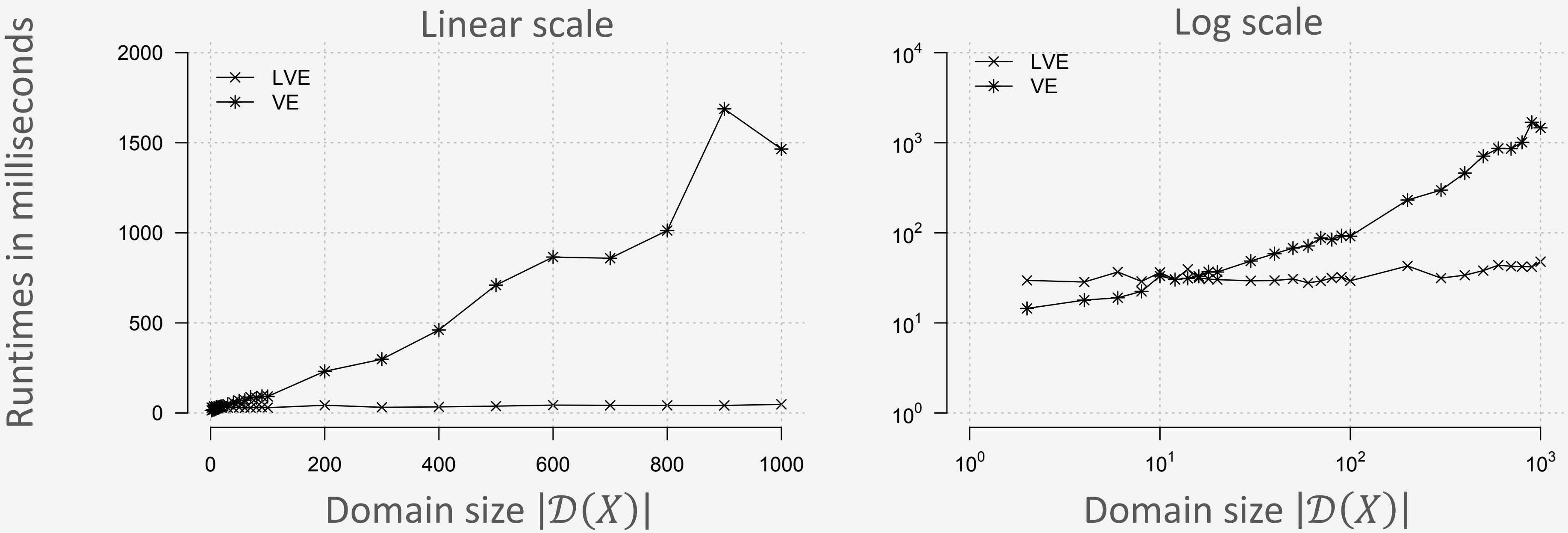
```
type Person;
guaranteed Person x[3];

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

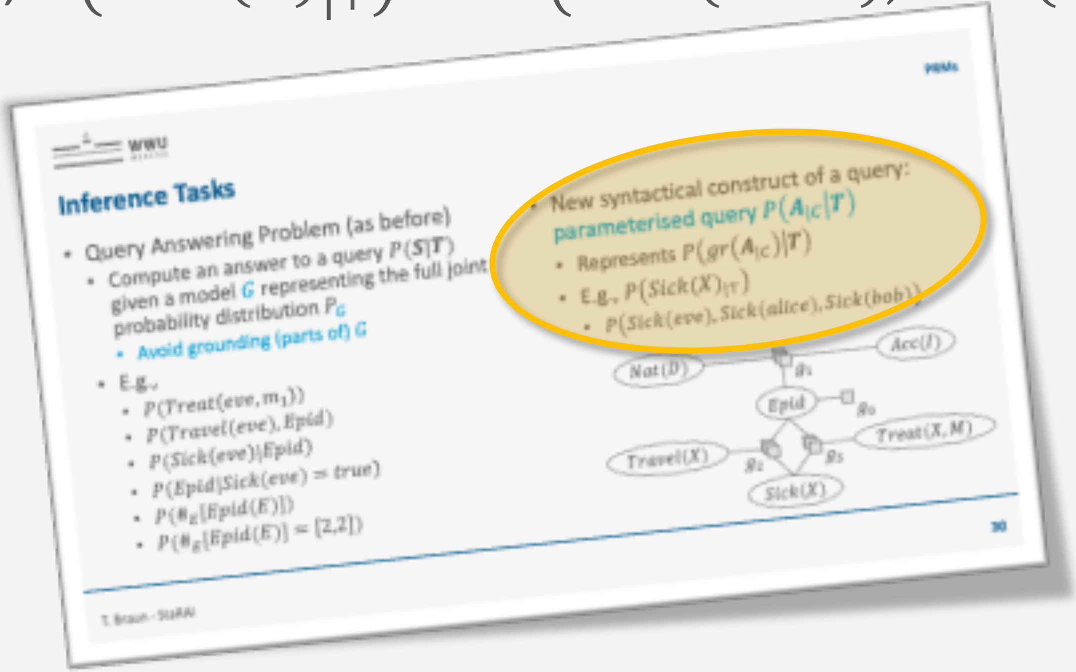


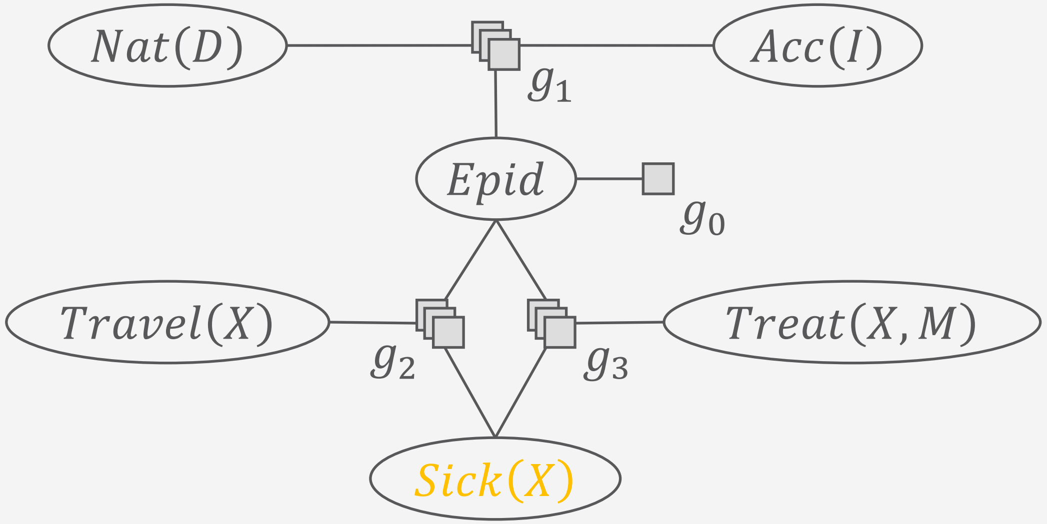
- Running example model with all domain sizes 2, except $|dom(X)| \in \{2,4,\ldots,20,\ 30,\ldots,100,200,\ldots,1000\}$
- Query: $P\big(Travel(x_1)\big)$



What About Parameterised Queries?

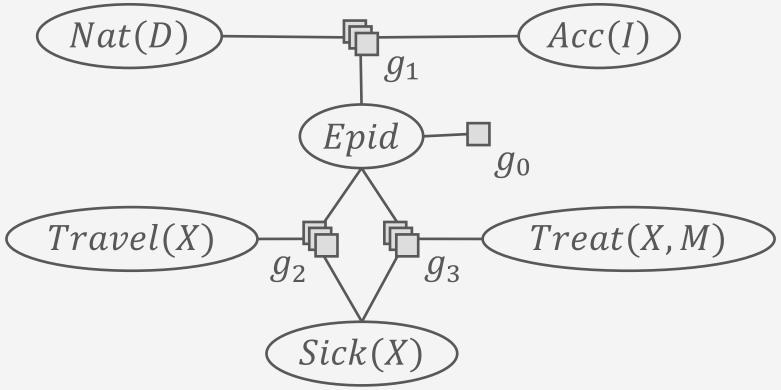
- Logical variables allowed in query terms: $P(\boldsymbol{A}_{|C} | \boldsymbol{T})$
  - Represents a conjunctive query $P(gr(\boldsymbol{A}_{|C}) | \boldsymbol{T})$
- E.g., $P(Sick(X)_{|\top})$ for $P(Sick(alice), Sick(eve), Sick(bob))$



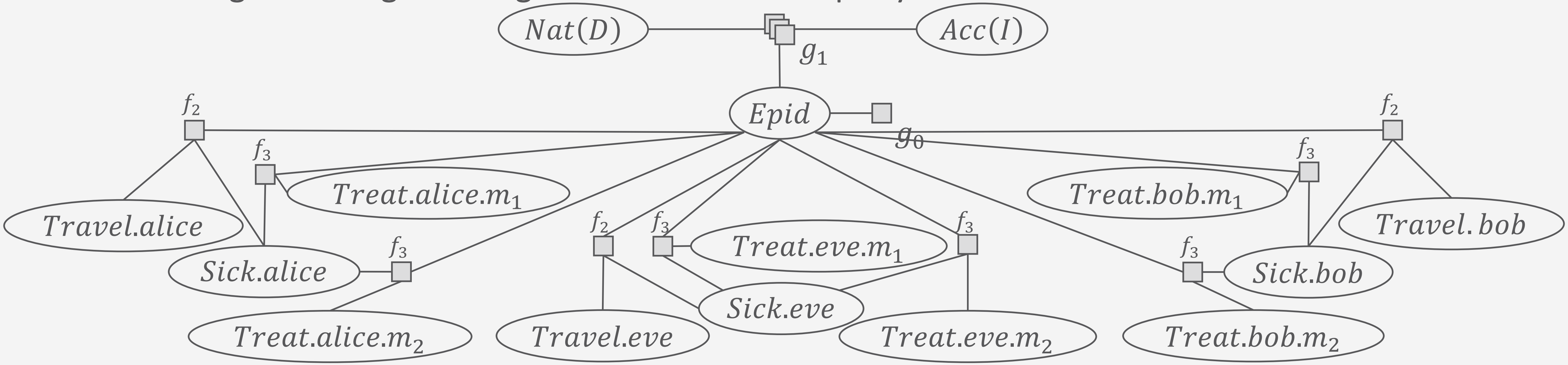


Tanya Braun and Ralf Möller: Parameterised Queries and Lifted Query Answering. In: *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

Indistinguishable Query Terms

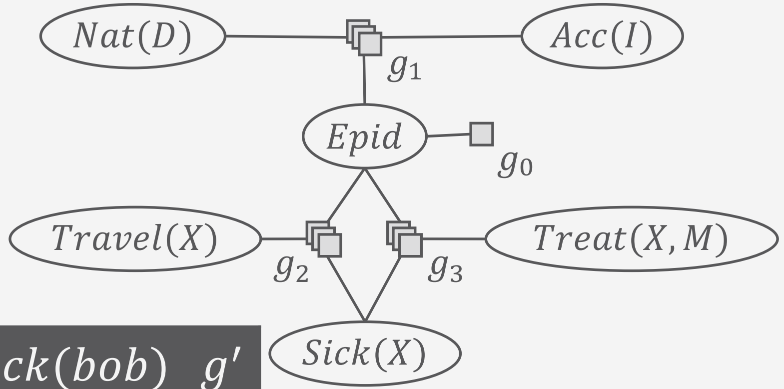


- Indistinguishable instances in query:
  - $P(Sick(alice), Sick(eve), Sick(bob))$
- Standard LVE:
  - Shattering leads to groundings w.r.t. constants in query



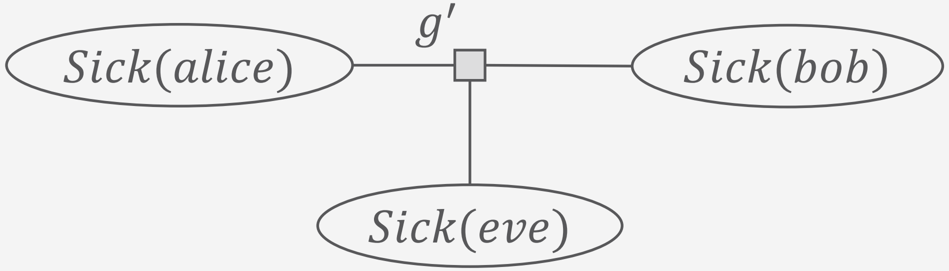
… And Their Effect

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



$Sick(alice)$	$Sick(eve)$	$Sick(bob)$	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

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



LVE for Parameterised Queries

- To avoid grounding, take PRV representation of query terms and apply LVE as before
  - Shatter model on constraint of query terms → maximum of two groups per parfactor and logical variable
  - Eliminate all non-query terms
    - If logical variable prevents application of operator → count or ground logical variable from query
  - After all non-query terms eliminated, count or ground remaining logical variables to make logical variables explicit
    - Otherwise only in representative form but not a joint over all groundings
- At the end, the result contains the logical variables counted (or grounded)
  - If counting the logical variables of $\boldsymbol{A}_{|C}$ is not possible, then LVE needs to ground them to ensure a distribution over $\boldsymbol{A}_{|C}$

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

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

Normalisation changes to account for histograms encoding multiple assignments

$\mathbf{LVE}(G, \boldsymbol{Q}_{|C}, \{g_e\}_{e=1}^m)$
  $G \leftarrow$ 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 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 on some parfactors in G
  $g \leftarrow$ Multiply all parfactors in G into one parfactor
  $g \leftarrow$ *Normalise* the potentials in g
  return g

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

- Histogram h may encode multiple assignments $\{\boldsymbol{a}_i\}_{i=1}^{Mul(h)}$
  - $Mul(h)$ assignments have the potential $\phi(h)$
  - Incorporate into normalisation (just like we needed to incorporate that into SUM−OUT)
- To get the probability of one assignment $\boldsymbol{a}$ behind histogram h in parfactor $\phi(\#_X[R(X)])$:

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

  - Probability of $Mul(h)$ assignments

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

  - Distribution:

$$\sum_{h \in ran(\#_X[R(X)])} p_h = \sum_{h \in ran(\#_X[R(X)])} Mul(h) \cdot p_{\boldsymbol{a}|h} = 1$$

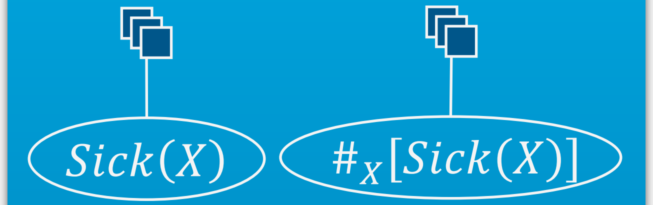
$\#_X[Sick(X)]$	g
[0,3]	1
[1,2]	2
[2,1]	3
[3,0]	4

Tanya B. Rescued from a Sea of Queries: Exact Inference in Probabilistic Relational Models. PhD Thesis, 2020.

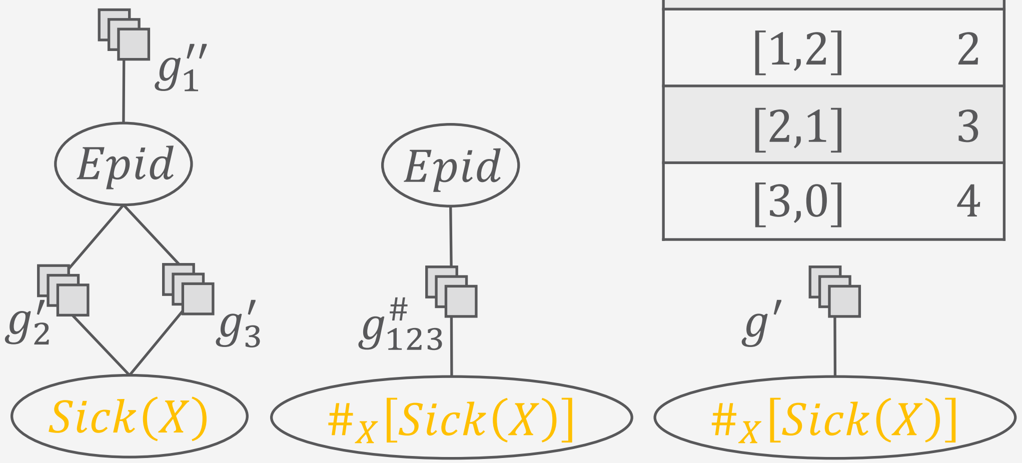
Example

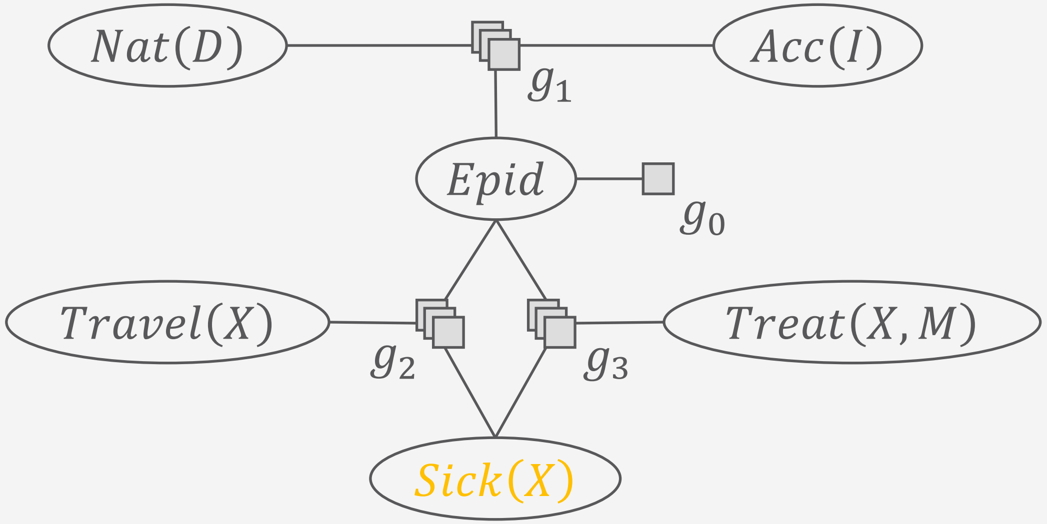
- Query: $P(Sick(X))$
- No shattering necessary with ⊤ constraints
- Elimination:
  - Eliminate $Treat(X, M)$
  - Eliminate $Travel(X)$
  - Count-convert $Acc(I)$
  - Eliminate $Nat(D)$
  - Eliminate $\#_I[Acc(I)]$
  - Eliminate $Epid$
    - Multiply parfactors (fulfil precondition 1)
    - Count-convert X (fulfil precondition 2)
    - Sum out $Epid$

Here, count conversion as part of elimination, but if logical variables remaining after elimination, count conversions afterwards (trivially possible):



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





Example

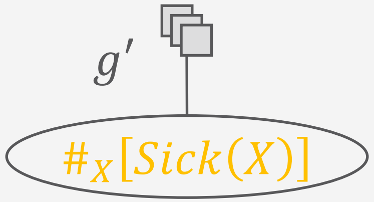
- Query: $P\big(Sick(X)\big)$
- Elimination: Finished
- Normalisation:

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

  - Probability distribution:

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

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



$\#_X[Sick(X)]$	ϕ'	single assignment	all assignments
[0,3]	1	$\rightarrow {}^{1}/_{20}$	$\rightarrow {}^{1}/_{20}$
[1,2]	2	$\rightarrow {}^{2}/_{20}$	$\rightarrow {}^{6}/_{20}$
[2,1]	3	$\rightarrow {}^{3}/_{20}$	$\rightarrow {}^{9}/_{20}$
[3,0]	4	$\rightarrow {}^{4}/_{20}$	$\rightarrow {}^{4}/_{20}$

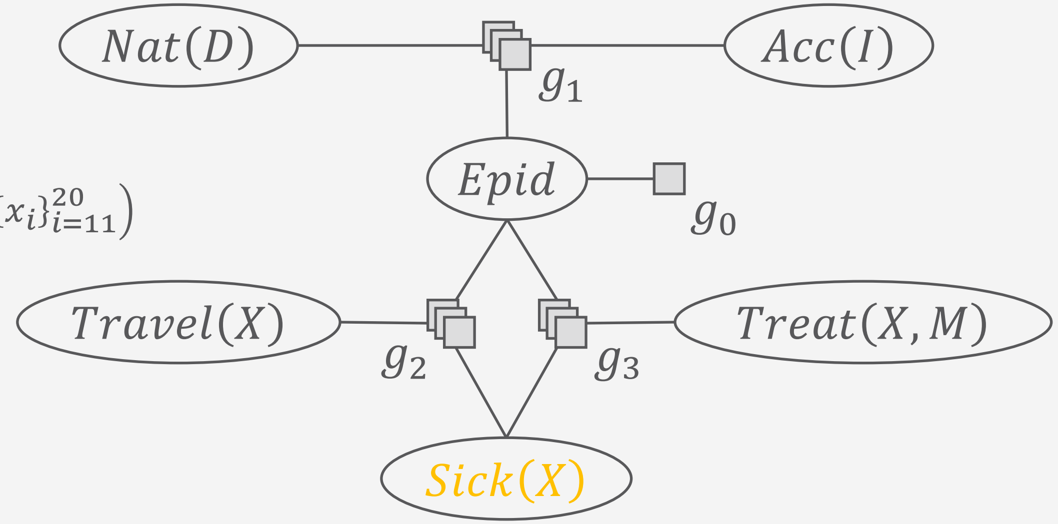
Splits Affecting Query Logical Variables

- Logical variables $\boldsymbol{X}$ in query terms may be split (or grounded) in result
  - If splits of the model affect the query logical variables
- Prominent case: evidence; three cases given query PRV $R(\boldsymbol{X})_{|C}$, evidence PRV $E(\boldsymbol{Y})_{|C_e}$
  1. Overlap of instances (i.e., $R(\boldsymbol{X}) = E(\boldsymbol{Y})$): $gr\left(R(\boldsymbol{X})_{|C}\right) \cap gr\left(E(\boldsymbol{X})_{|C_e}\right) \neq \emptyset$
     - Split C on the overlap with C_e, i.e., $C/_{\boldsymbol{X}}C_e$ → instances of C_e will be absorbed
     - Result has $R(\boldsymbol{X})_{|C}$ partitioned into $C/_{\boldsymbol{X}}C_e \setminus C_e$ (absorbed instances: probability 1 of observed value)
  2. Overlap of constants ($\boldsymbol{Z}$ shared logical variables): $gr\left(\pi_{\boldsymbol{Z}}(\boldsymbol{X}_{|C})\right) \cap gr\left(\pi_{\boldsymbol{Z}}(\boldsymbol{Y}_{|C_e})\right) \neq \emptyset$
     - 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 (different evidence applies)
  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 (maybe partitioned for other reasons)

Tanya B. Rescued from a Sea of Queries: Exact Inference in Probabilistic Relational Models. PhD Thesis, 2020.

Splits Affecting Query Logical Variables: Examples

- Given query $P\left(Sick(X)_{|(X,\{x_i\}_{i=1}^{20})}\right)$:
  1. Overlap of instances: evidence $\phi_e\big(Sick(X)\big)_{|(X,\{x_i\}_{i=1}^{10})}$
     - Result: $\phi(\#_X[Sick(X)])_{|(X,\{x_i\}_{i=11}^{20})}$
  2. Overlap of constants: evidence $\phi_e\big(Travel(X)\big)_{|(X,\{x_i\}_{i=1}^{10})}$
     - Result: $\phi(\#_{X'}[Sick(X')], \#_{X''}[Sick(X'')])_{|((X',X''),\{x_i\}_{i=1}^{10}\times\{x_i\}_{i=11}^{20})}$
  3. No overlap: evidence $\phi_e\big(Nat(D)\big)_{|(D,\{d_i\}_{i=1}^{2})}$
     - Result: $\phi(\#_X[Sick(X)]\ )_{|(X,\{x_i\}_{i=1}^{20})}$



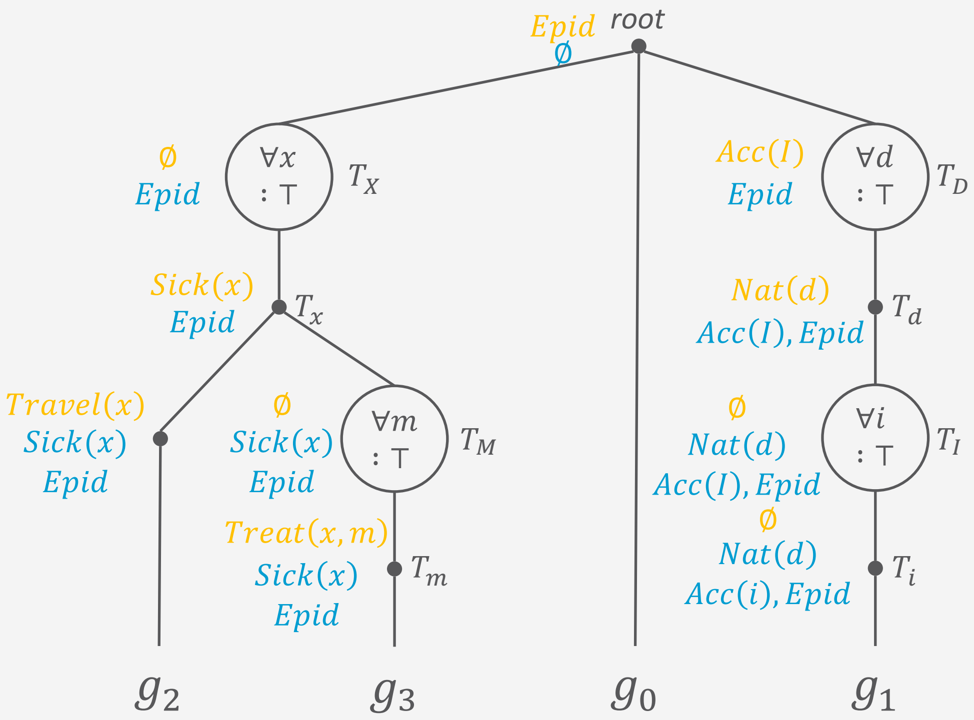
Tanya B. Rescued from a Sea of Queries: Exact Inference in Probabilistic Relational Models. PhD Thesis, 2020.

Interim Summary

- LVE lifted operators
  - Eliminate PRVs: lifted summing out, lifted absorption
  - Enable elimination: lifted multiplication, count conversion, splitting
    - Other based on splitting: expand, ground, count-normalise
- Shattering = splitting on query terms, evidence, model constraints
  - Pre-emptive, on-demand
- LVE algorithm
  - Heuristics
  - Implementation
  - Version for parameterised queries
    - Effect of evidence: possibly partitioned result

Theoretical Analysis

Lifted Variable Elimination



Runtime Complexity of Probabilistic Inference Using PGMs

- *Informal*
  Worst-case size of interim result *times* number of eliminations
- Decomposition tree (*dtree*)
  - Data structure to characterise runtime complexity
  - Represents a VE run for a query
    - Most representative query: empty query $P(.)$, i.e., the normalisation constant
  - Acyclic tree with factors or interim results associated with nodes
    - Leaves: Factors of model
    - Inner nodes: interim results after an elimination of a variable
      - Root: final result (query answer)
      - Edge between an inner node T_i and a child node T_j if factor / interim of T_j involved in elimination of variable
      - If variable appears in more than one factor, then more than one child

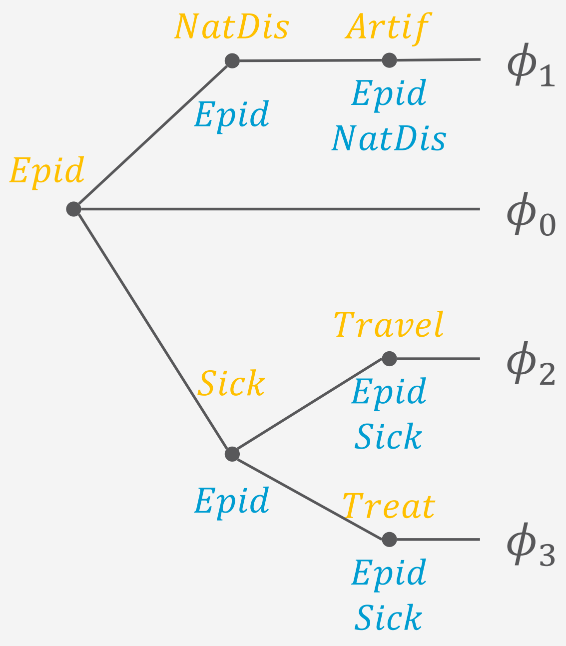
Without actually realising the interim results, a dtree allows for determining a worst-case size based on the variables involved

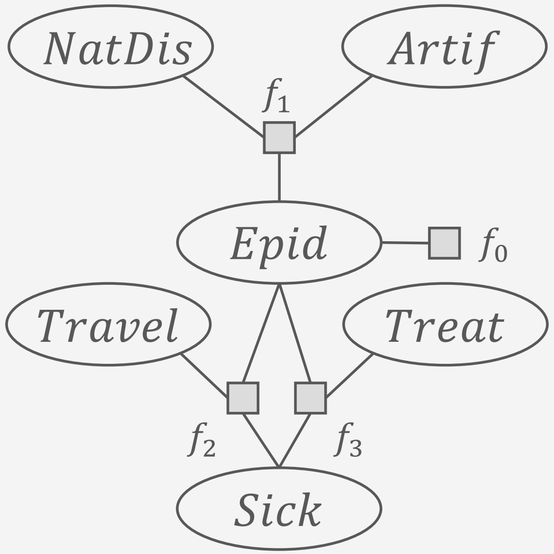
More details: Adnan Darwiche, “Modelling and Reasoning with Bayesian Networks” (2009), Chs. 8.3 + 9.5

Example: Dtree – Bottom-up Construction as VE Representation

$$P(.) = \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a)$$

$$\sum_{s \in \mathrm{Val}(S)} \sum_{tr \in \mathrm{Val}(Tr)} \phi_2(tr,e,s) \sum_{tt \in \mathrm{Val}(Tt)} \phi_3(e,s,tt)$$

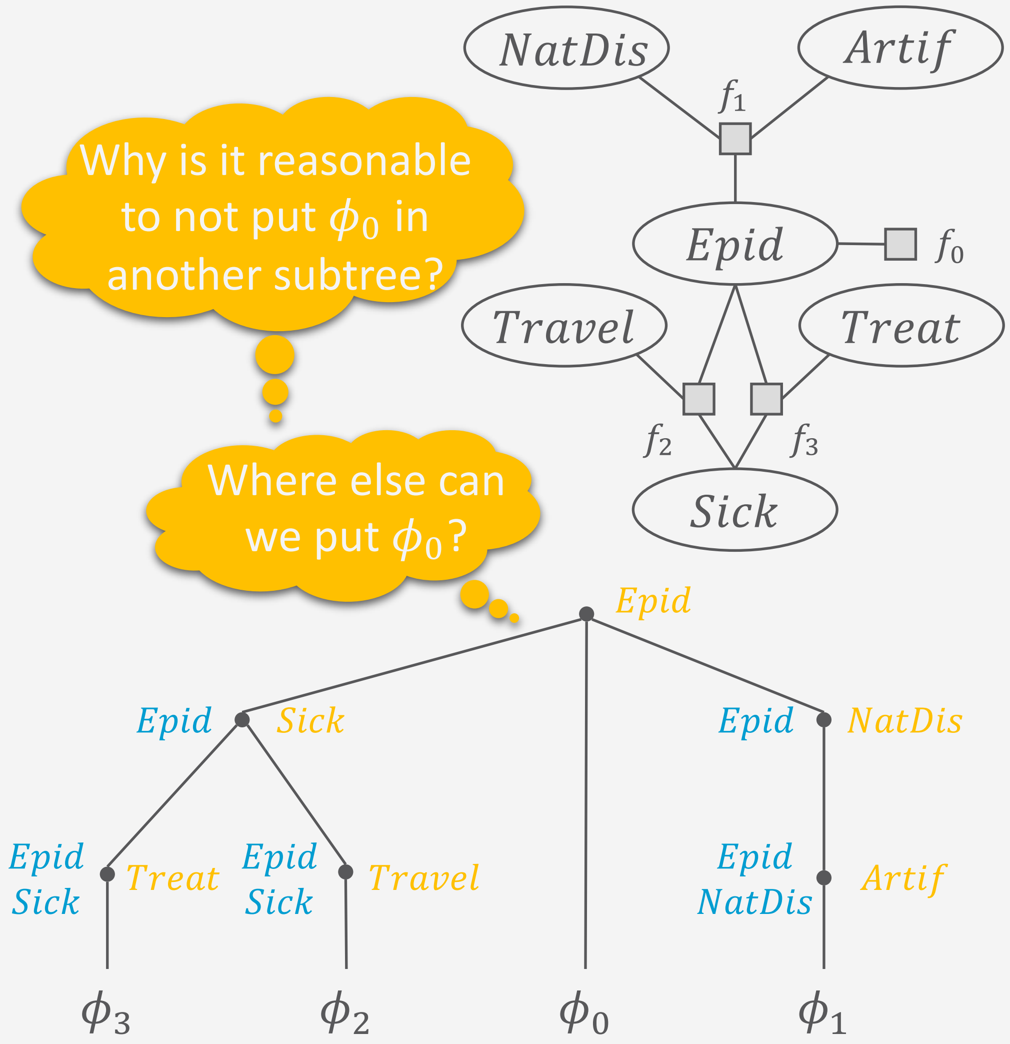




Computations in different subtrees can be parallelised, as they are independent from each other

Example: Dtree – Top-down Interpretation

- At beginning: Root node with model $F = \{f_i\}_{i=1}^{n}$ as current model F'
- Recursively partition F' at node k such that
  - Each partition $F_i \subseteq F'$ contains random variables $\boldsymbol{U}_i$ that do not appear in other partitions
    - Maximise size of $\boldsymbol{U}_i$ over all partitions
    - $\boldsymbol{U}_i$ can be eliminated without considering factors of other partitions
- For each partition F_i, add a child node i to k with F_i as current model F'
- Stop at a node if current model F' contains only one factor



Cutset, Context, Cluster

- Cutset
  - What gets eliminated at this node (*cut from the model*)

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

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

$\mathrm{rv}(T)$ returns the random variables, which appear in the factors of this subtree

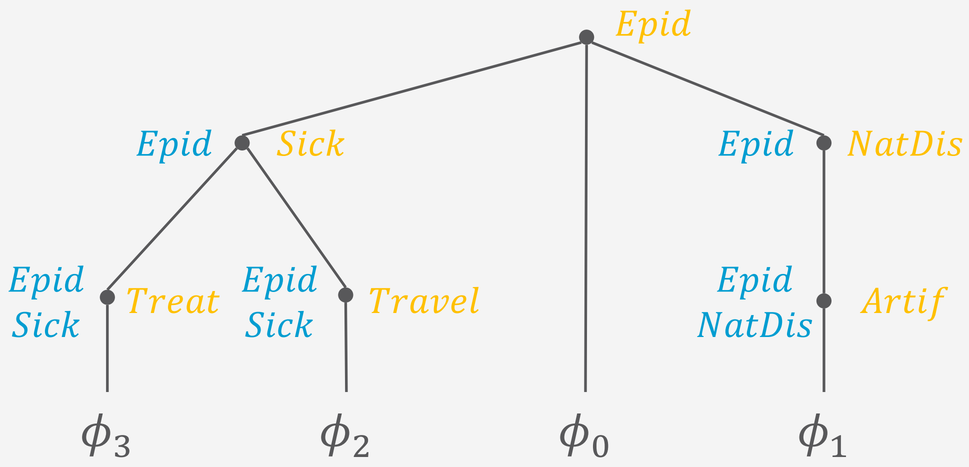
- Context
  - What is set during elimination (what else appears)

$$\mathrm{context}(T) = \mathrm{rv}(T) \cap \mathrm{acutset}(T)$$

- Cluster
  - Cutset and context together

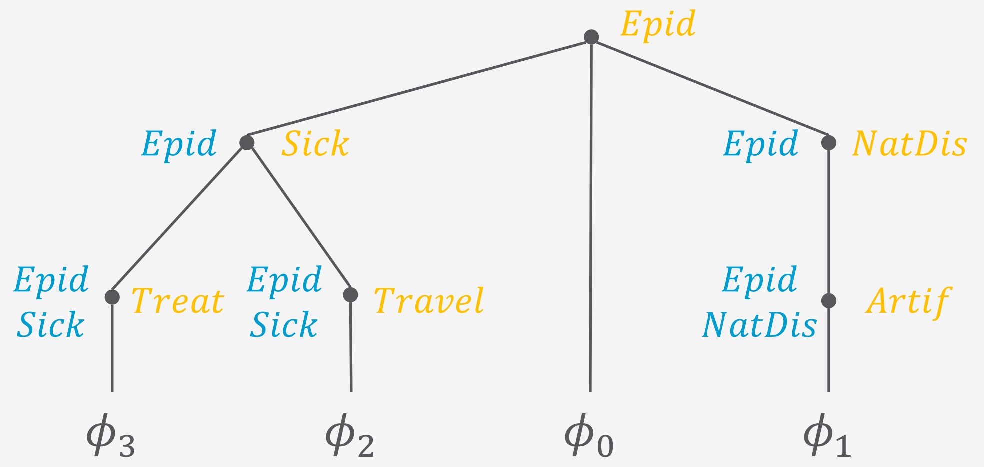
$$\mathrm{cluster}(T) = \mathrm{cutset}(T) \cup \mathrm{context}(T)$$

    - If T is a leaf, then $\mathrm{cluster}(T) = \mathrm{context}(T) = \mathrm{rv}(f)$



Cutset, Context, Cluster

- Largest cluster in tree T = *tree width w*
$$w = \max_{T_i \in \mathrm{Desc}(T)} |\mathrm{cluster}(T_i)|$$
- Induces a worst-case factor size
  - Cluster specifies, which random variables involved in an elimination
    - Appear together in a factor
  - Largest cluster → largest number of arguments of a factor
  - Example:
    - $w = 3$, worst-case factor size $2^w = 2^3$
- w bounded from below by largest input factor size:
$$w \geq m = \max_{f \in F} |\mathrm{rv}(f)|$$
  - When learning a model, avoid factors with many arguments (e.g., bound degree in FG / MN)

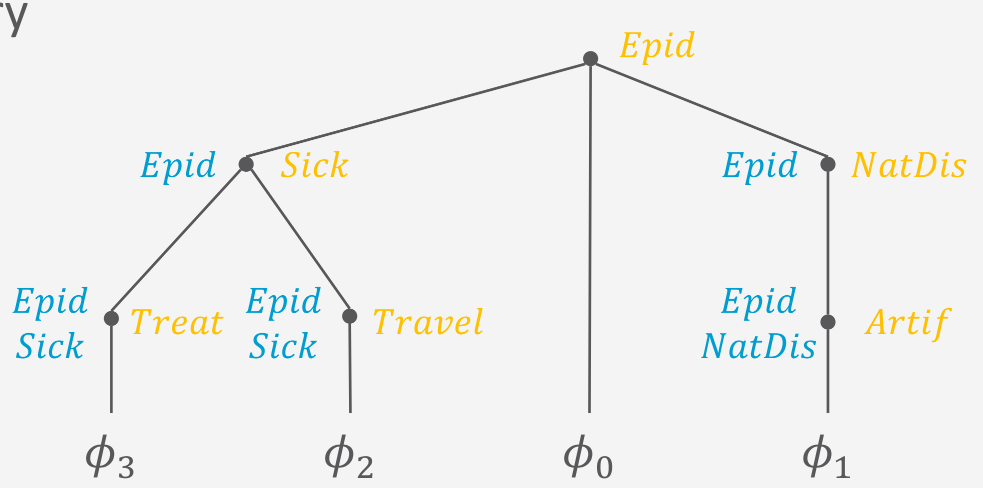


Back to Runtime Complexity of Probabilistic Inference in PGMs

- *Informal*
  Worst-case size of interim result *times* number of eliminations
- Decomposition tree (*dtree*)
  - Tree width w = worst-case number of arguments
  - Number of inner nodes n_T = Number of eliminations $\leq |rv(F)|$
    - $n_T = |rv(F)|$ as upper bound, i.e., asking the empty query
- *Formal*:
  Runtime complexity of VE

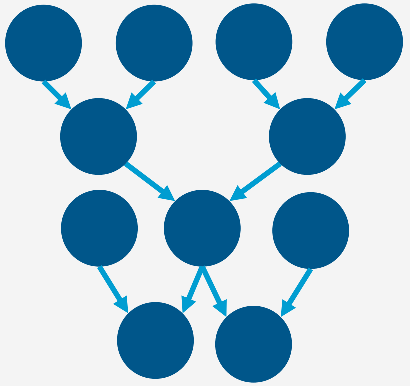
$$O(n_T \cdot r^w)$$

  - $r = \max_{R \in rv(F)} |ran(R)|$
  - Compare with inference using full joint P_F: $O(r^{n_T})$

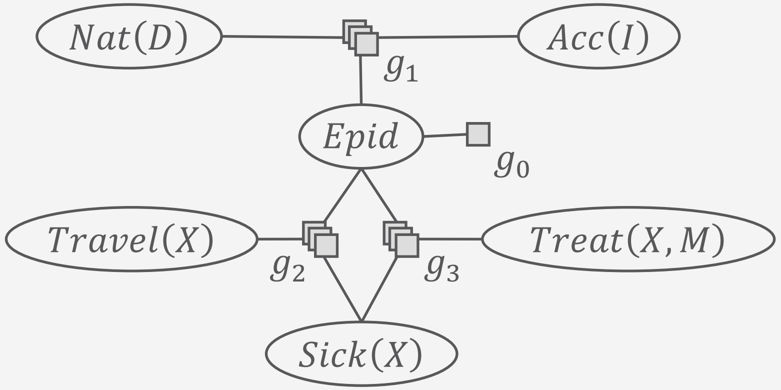


Complexity and Tractability

- Query answering problem is tractable IF
  - it is solved by an efficient algorithm in time *polynomial* w.r.t. the number of random variables
- Query answering problem in general is intractable
  - No guarantees that $w \ll n_T$
- Exceptions make assumptions about model structure
  - E.g., *polytree* BNs B
    - Directed graph with $P(R|pa(R))$ at each node R
    - Ensures that $w = \max_{R \in rv(B)} |pa(R)| + 1$
    - Also holds for tree-shaped FGs and their MN representation
    - Assumes that degree is not in order of n_T
  - E.g., *PRMs* → *how? Upcoming…*



Polytree
(no cycles in undirected version)



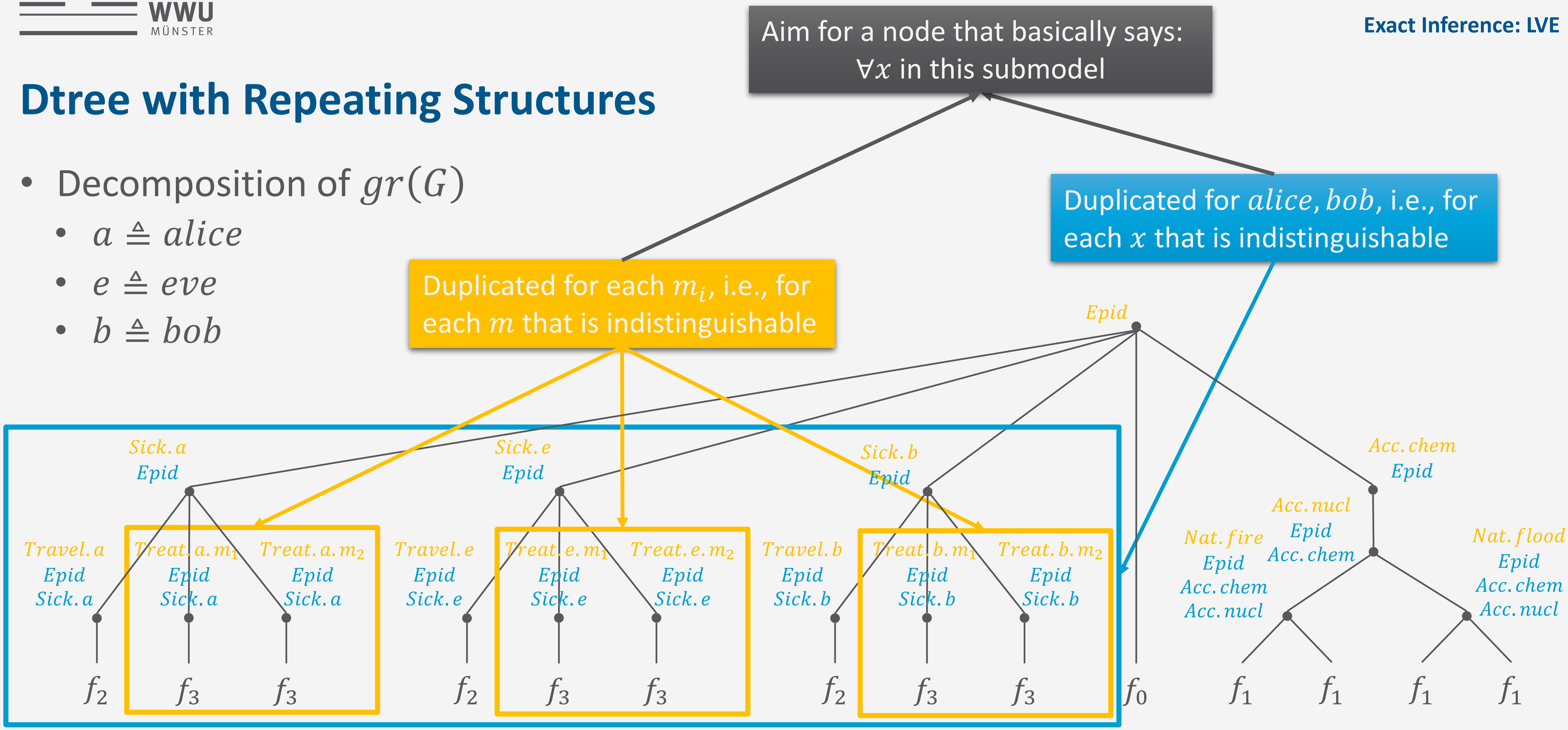
Complexity of Probabilistic Inference in PRMs

- *Informally*, LVE complexity still
  worst case size of an interim result $times$ number of eliminations

- Use a so-called first-order dtree (FO dree), to get worst case size of an interim result characterised by so-called lifted width
  - In dtree representation of VE for $gr(G)$,
    duplicate subtrees whenever a lifted sum-out applicable in G
  - In FO dtree of LVE for G,
    representative subtree for lifted sum-out (compactly encode duplicate subtrees)

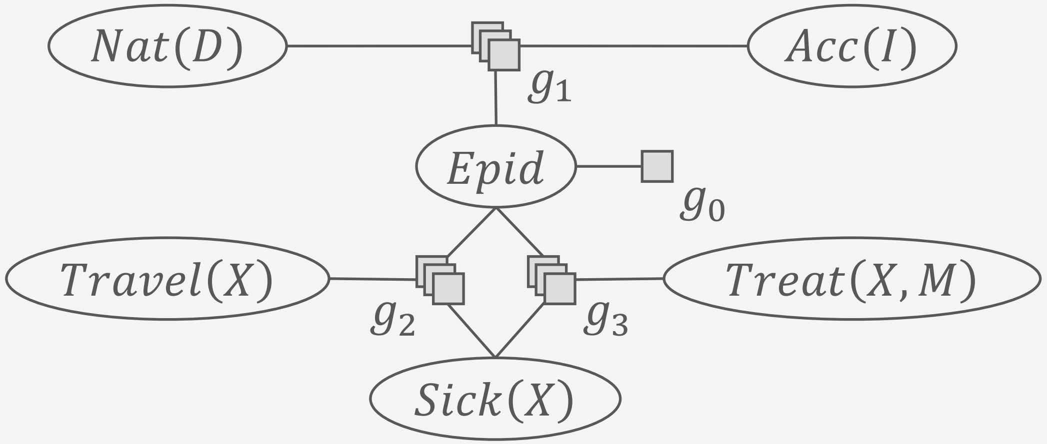
Dtree with Repeating Structures

- Decomposition of $gr(G)$
  - $a \triangleq alice$
  - $e \triangleq eve$
  - $b \triangleq bob$



Decomposition into Partial Groundings

- Introduce a new inner node: DPG node denoted $\forall x : C$
  - DPG = Decomposition into partial groundings
    - Replaces a logical variable with a representative constant
    - The resulting model/subtree is identical for each constant represented
      - Allows for considering the resulting model without the grounded logical variable for further decomposition (top-down)
  - E.g., submodel below $Epid$ in the graph
    - Logical variable X appears in each parfactor
    - Grounding X leads to copies of the same submodel
    - Replace X with representative x → 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
  * possible to rewrite any model in polynomial time into normal form
  - Domains are either disjoint or identical
  - Logical variables share the same name if they refer to the same domain over different parfactors
  - Constraints are ⊤
    - Formal definition by Taghipour includes inequality constraints

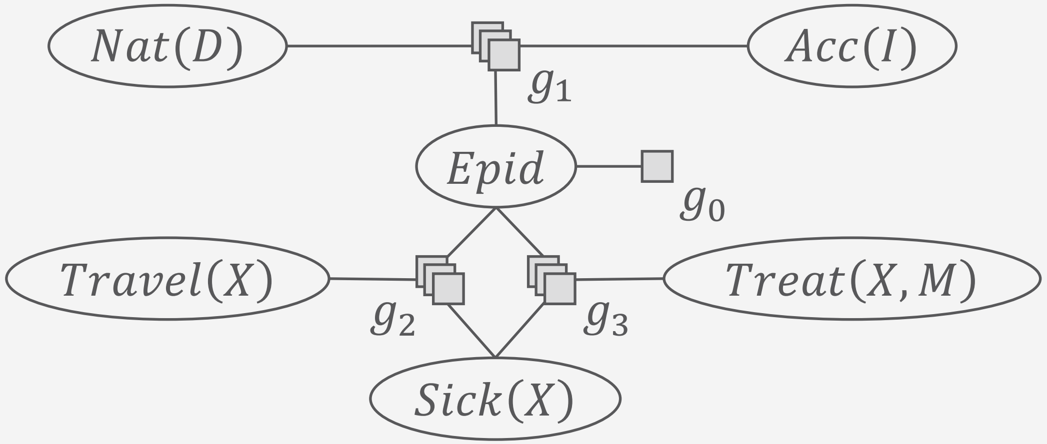
  ⇒ No further splitting operations necessary (split, expand, count-normalise)

- Decomposition into partial groundings of G by logical variable X with $\forall g \in G: X \in lv(g)$

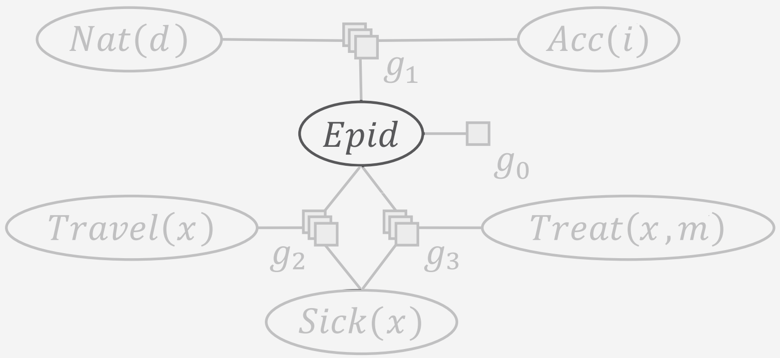
$$DPG(G, X) = \bigcup_{g \in G} g\theta\ , \theta = \{X \rightarrow x\}$$

FO Dtree Construction

- Recursively, starting with G as the current model G' at the root
  - Check if there exists logical variable 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 $\theta = \{X \rightarrow x\}$ 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' on logical variables (if exist) or random variables into $\{G_i'\}_{i=1}^n$
    - Add a child node for each G_i' with G_i' as current model
- Until
  - All logical variables replaced by representatives and
  - Only one parfactor per partition

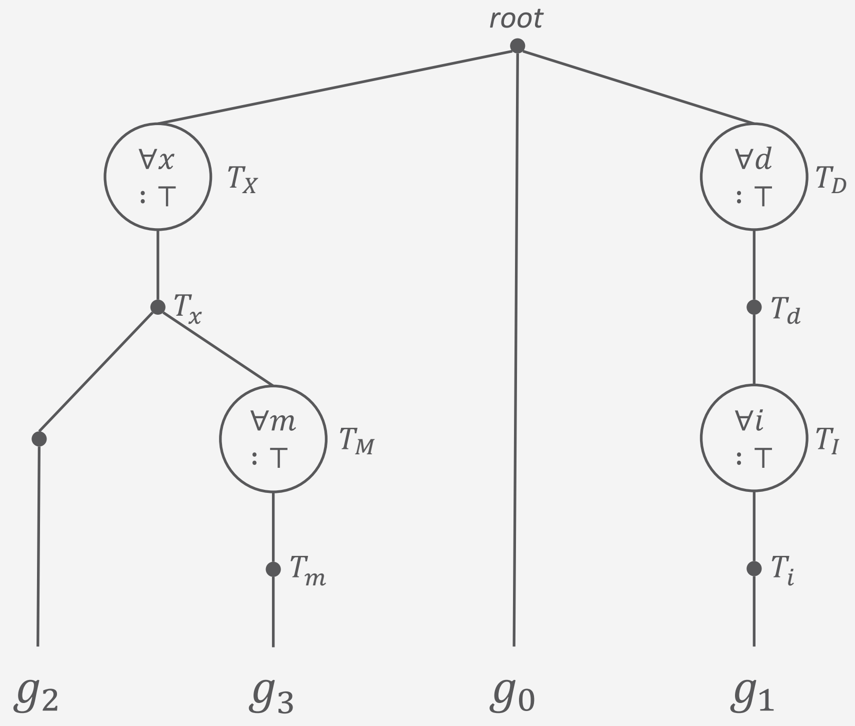


FO dtree: Example



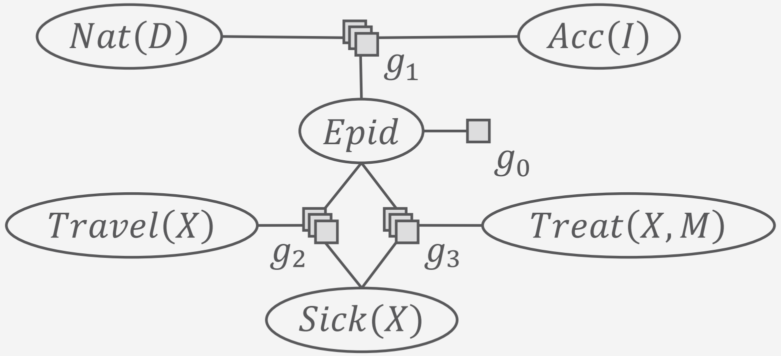
Root: In G, no logical variable for a DPG

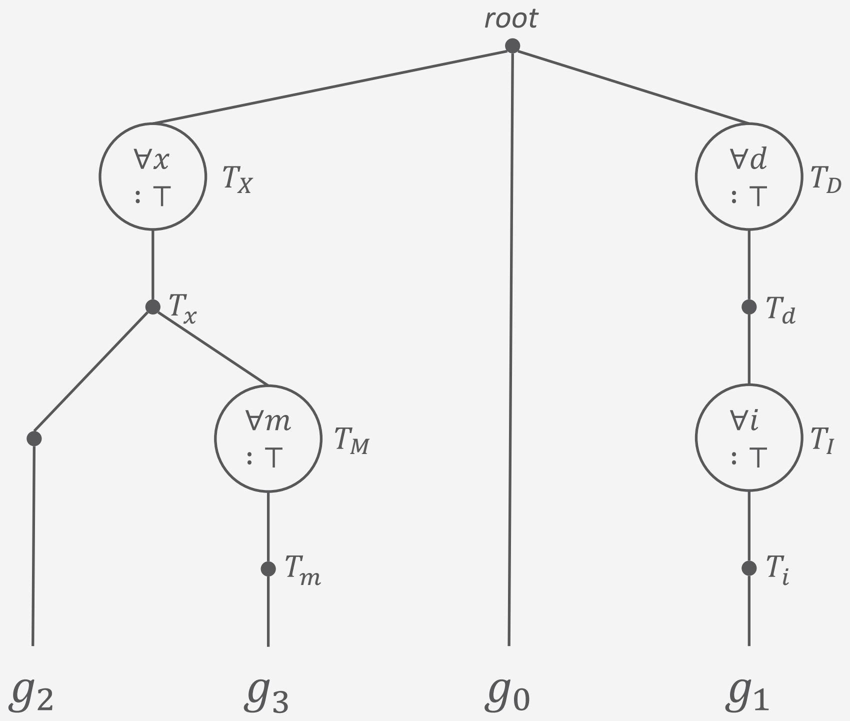
- Partition based on, e.g., $X \rightarrow G_0 = \{g_2, g_3\}, G_1 = \{g_1\}, G_2 = \{g_0\}$
- Left: $G_0 = \{g_2, g_3\}$
  - DPG with X → Replace X with x
  - No logical variable for DPG
  - Partition based on $M \rightarrow G_{01} = \{g_2\}, G_{02} = \{g_3\}$
  - Left: $G_{01} = \{g_2\}$
    - No logical variables and only one parfactor left
  - Right: $G_{02} = \{g_3\}$
    - DPG with M → Replace M with m
      - No logical variables and only one parfactor left
- Right: $G_1 = \{g_1\}$
  - DPG with D → Replace D with d
  - DPG with I → Replace I with i
    - No logical variables and only one parfactor left



FO Dtree Definition

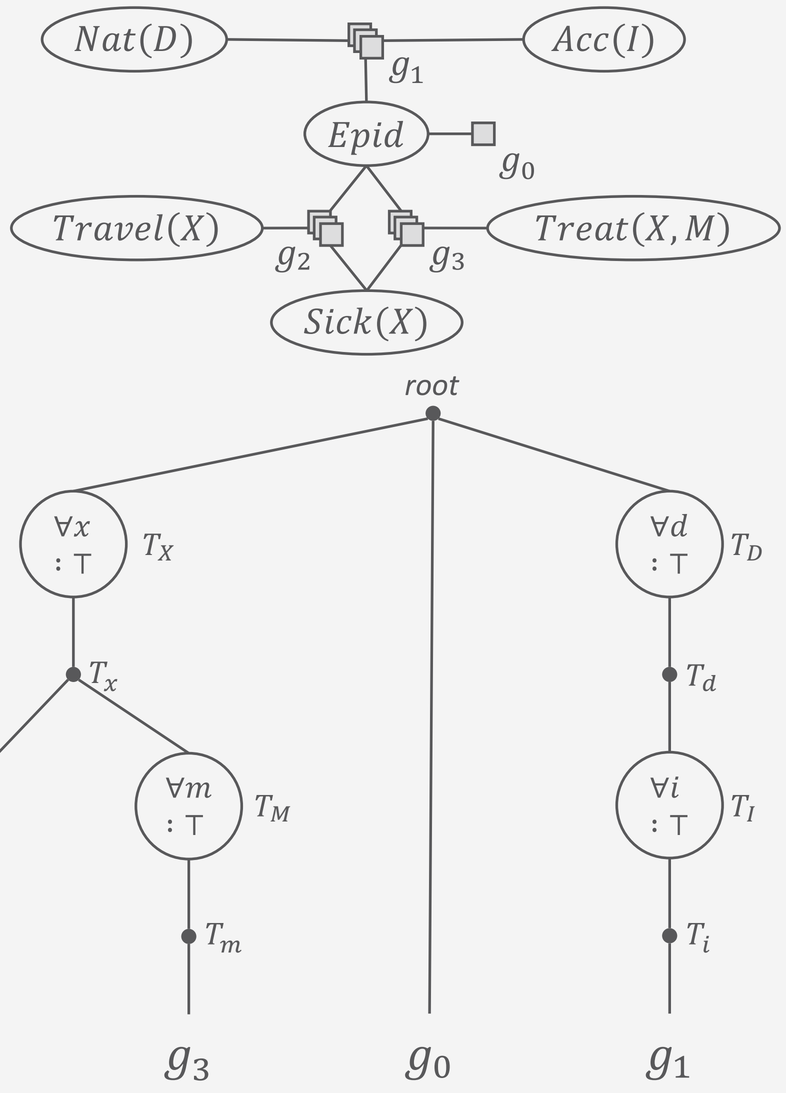
- An FO dtree has three node types
  - DPG node T_X
    - Represents a DPG (top-down)
    - Given by a tuple (X, x, C) with X a logical variable, x a representative constant, and C a constraint
      - In this lecture: $C = \top$
      - Denoted $(\forall x : C)$ in graphical representation of the tree
  - 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 each
- Then, an FO dtree T for a model G is given by $T = (V, E)$ where
  - $V = DPG \cup VE \cup Leaf$
  - $E = (DPG \times VE) \cup (VE \times DPG) \cup (VE \times VE) \cup (VE \times Leaf)$
    - VE can follow DPG / VE nodes, DPG / leaf can follow VE nodes
  - Each DPG node T_X has a child VE node T_x whose model 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 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 has to partition initial model G into one partition of parfactors containing only random variables and one partition of parfactors containing logical variables



FO Dtree Properties (as before)

- Cutset

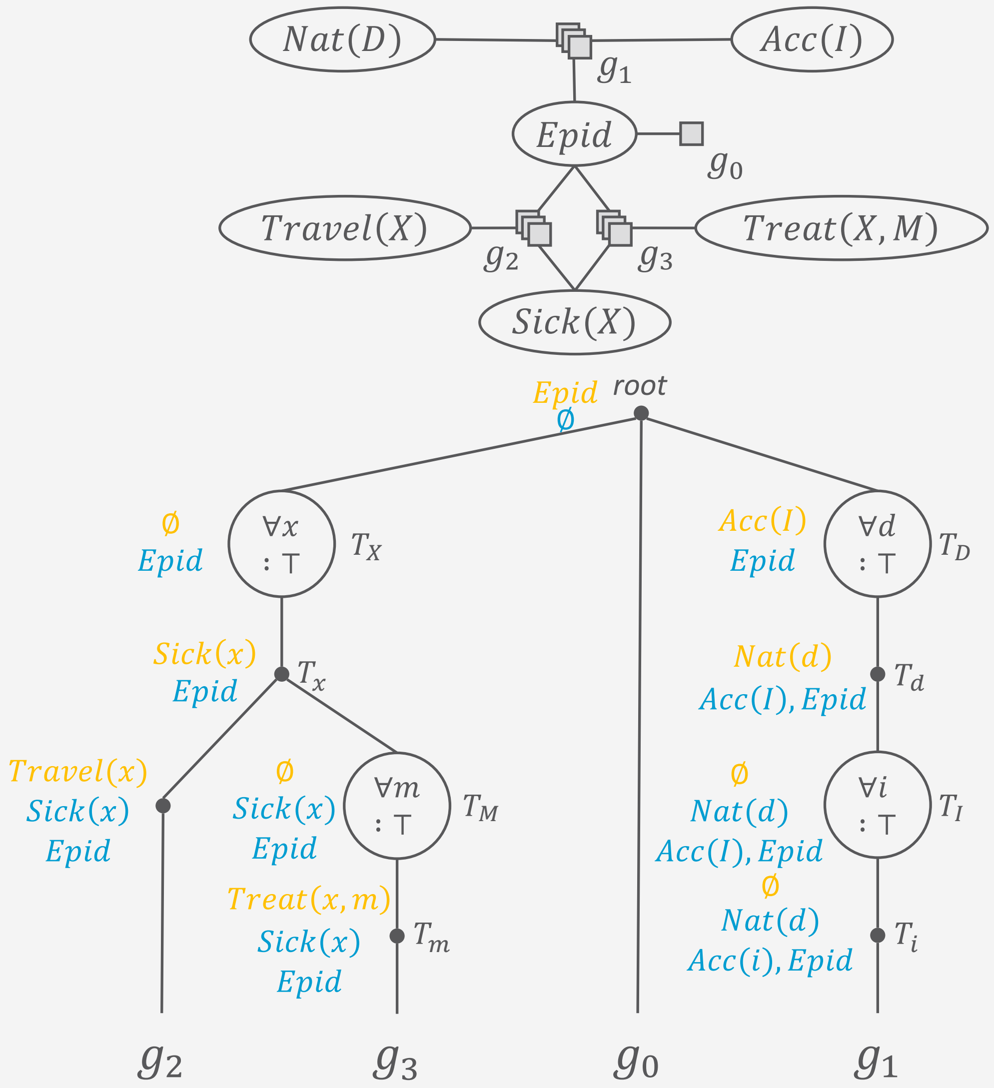
$$\mathrm{cutset}(T) = \left(\bigcup_{T_i, T_j \in \mathrm{Ch}(T)} \mathrm{rv}(T_i) \cap \mathrm{rv}(T_j)\right) \setminus \mathrm{acutset}(T)$$

  - Ancestor cutset: $\mathrm{acutset}(T) = \bigcup_{T' \in \mathrm{Anc}(T)} \mathrm{cutset}(T')$
  - Definitions of Ch if DPG nodes T_X involved
    - Θ_X all grounding substitutions of X
    - $\mathrm{Ch}(T_X) = \{T_{x\theta} | T_x \text{ is child of } T_X \wedge \theta \in \Theta_X\}$
    - $\mathrm{Ch}(T_y) = \{T_{X\theta} | T_X \text{ is child of } T_y \wedge \theta \in \Theta_X\}$
- Context

$$\mathrm{context}(T) = \mathrm{rv}(T) \cap \mathrm{acutset}(T)$$

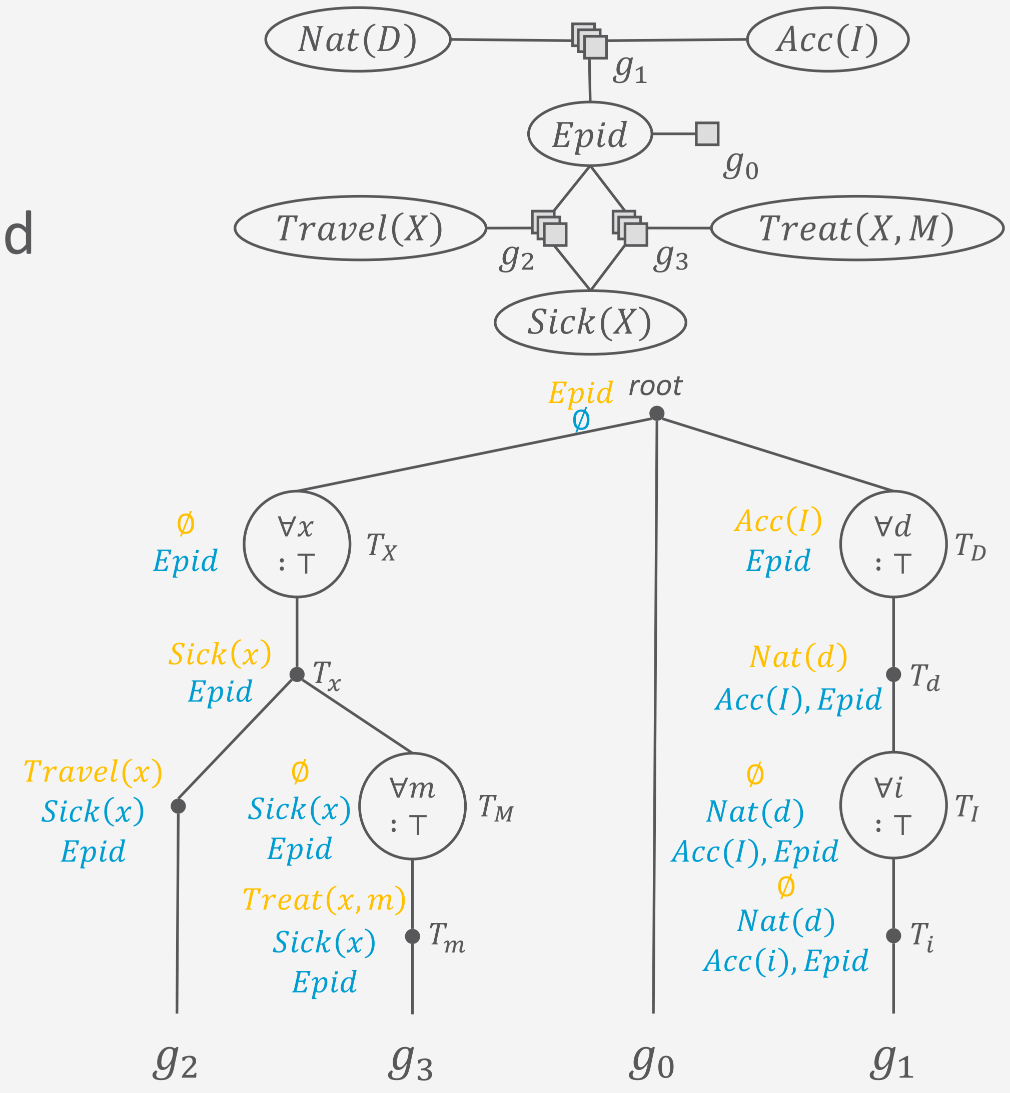
- Cluster

$$\mathrm{cluster}(T) = \mathrm{cutset}(T) \cup \mathrm{context}(T)$$



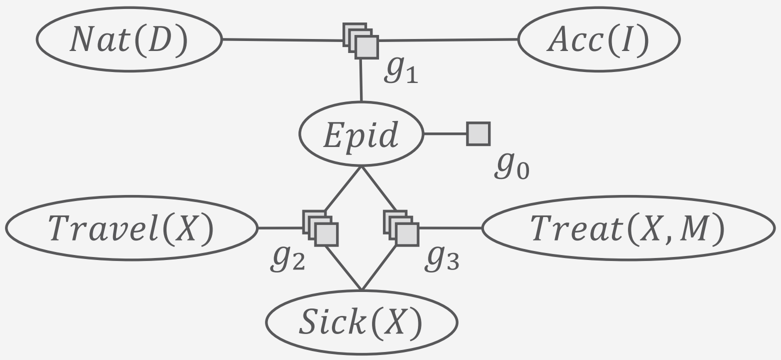
FO dtree: Bottom-up Interpretation

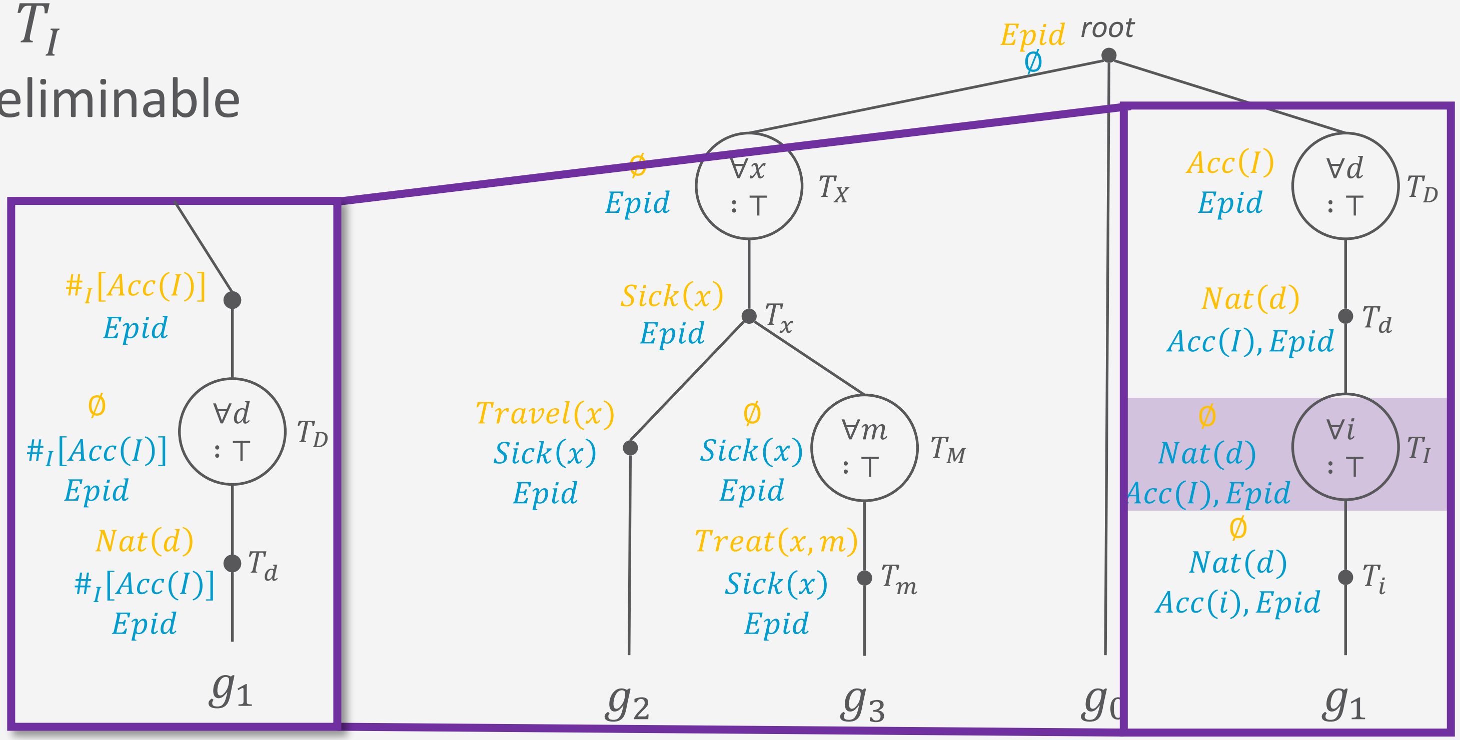
- If only lifted summing out and multiplication involved
  - VE node: (Multiplication), elimination
  - DPG node: Exponentiation
  - Cutset and context interpretation
    - Cutset of DPG child VE node:
      PRV summed out in representative summing out
    - Cutset of DPG node:
      Exponentiation for logical variable of DPG node
    - Context:
      All other PRVs involved at this point in operation



FO dtree: Bottom-up Interpretation

- If DPG logical variable occurs in the context of its DPG node: Count conversion necessary!
  - E.g., $Acc(I)$ in context of DPG node T_I
  - Shows "only" that PRV not directly eliminable
    - Occurs when eliminating $Nat(d)$ at T_d
    - No direct interpretation in terms of LVE operations
  - Rework FO dtree to represent calculations in count-converted model
    - E.g., consider model with $\#_I[Acc(I)]$ instead of $Acc(I)$



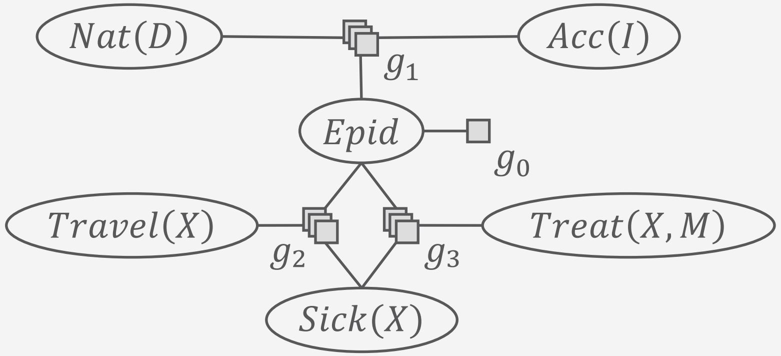


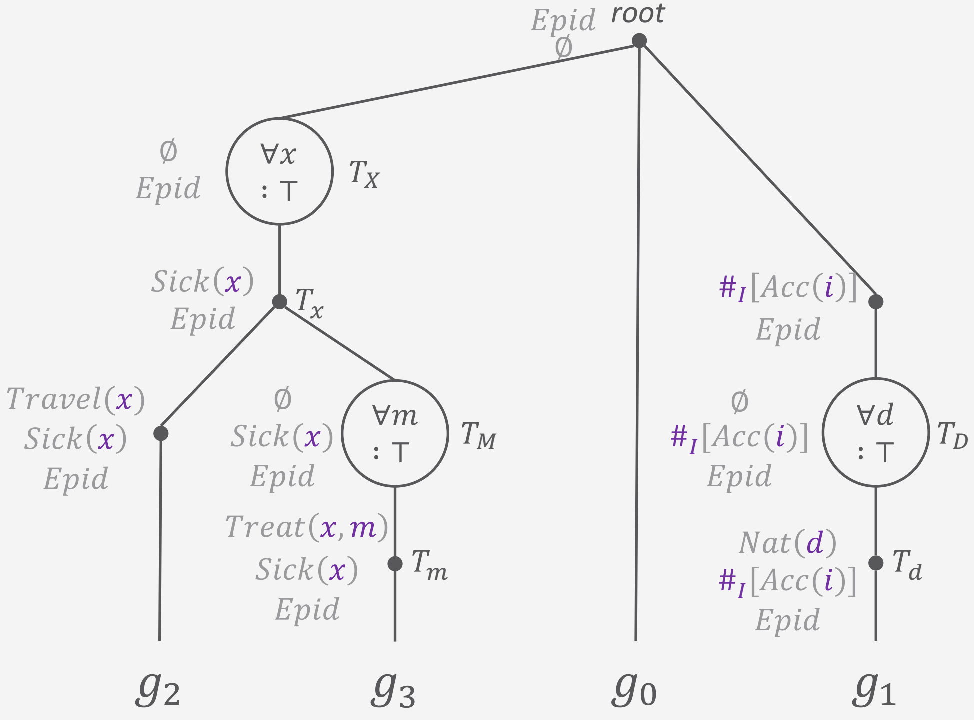
Liftability

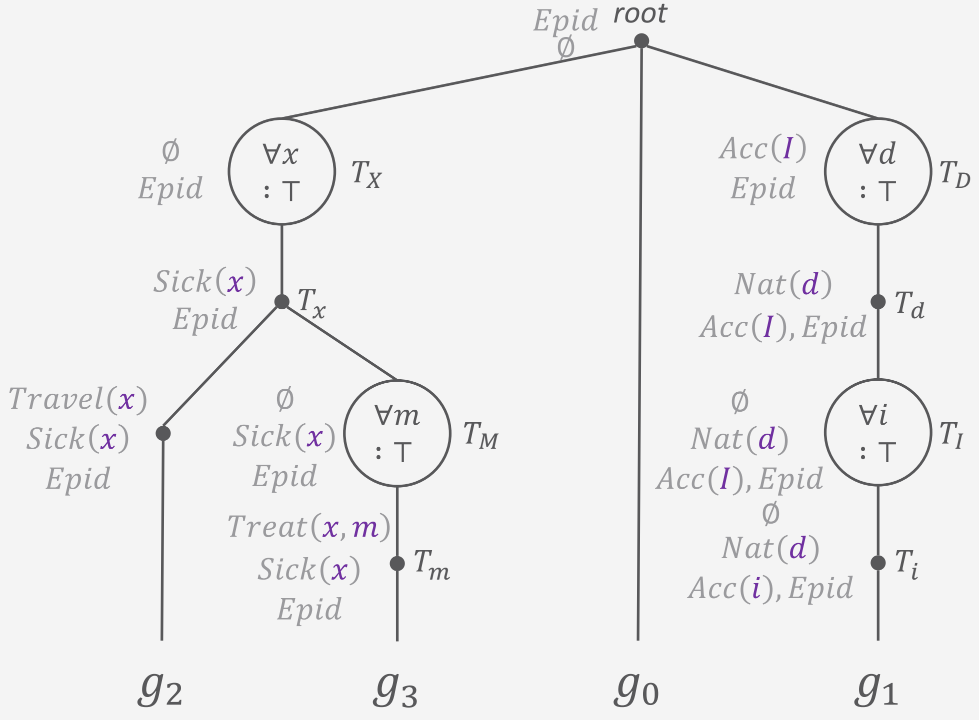
- Given an FO dtree T for a model G
  - If the clusters of T only consist of *PRVs with representative constants* and *PRVs with one logical variable*, then G has a lifted solution
    - Lifted solution: no groundings necessary; only lifted calculations ($\mathrm{sum-out}$, $\mathrm{multiply}$, $\mathrm{count-convert}$)
    - PRVs only with representative constants → lifted summing out possible
    - PRVs with one logical variable → count conversion necessary (and possible)
      - Called countable
  - T is called liftable
  - Apply the count conversions to the countable logical variables, 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

Liftability: Example FO Dtree

- Only PRVs with representative constants or one logical variable in the clusters → liftable
- If counting I and reworking the FO dtree → counted, liftable

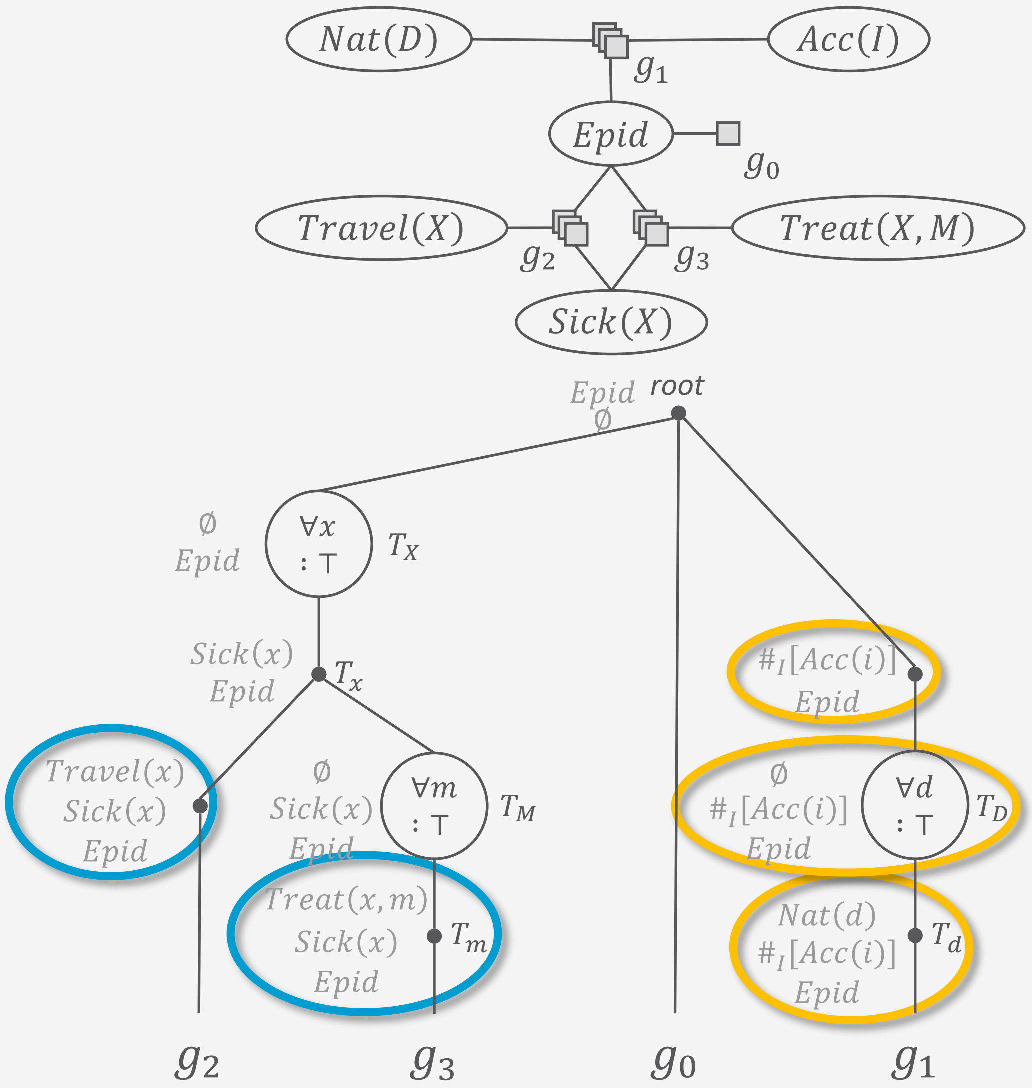






Lifted Width

- 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
  - $w_\#$ largest counting width
    - Largest number of CRVs in any cluster
  - E.g., $w_T = (w_g, w_\#)$ with $w_g = 3$, $w_\# = 1$



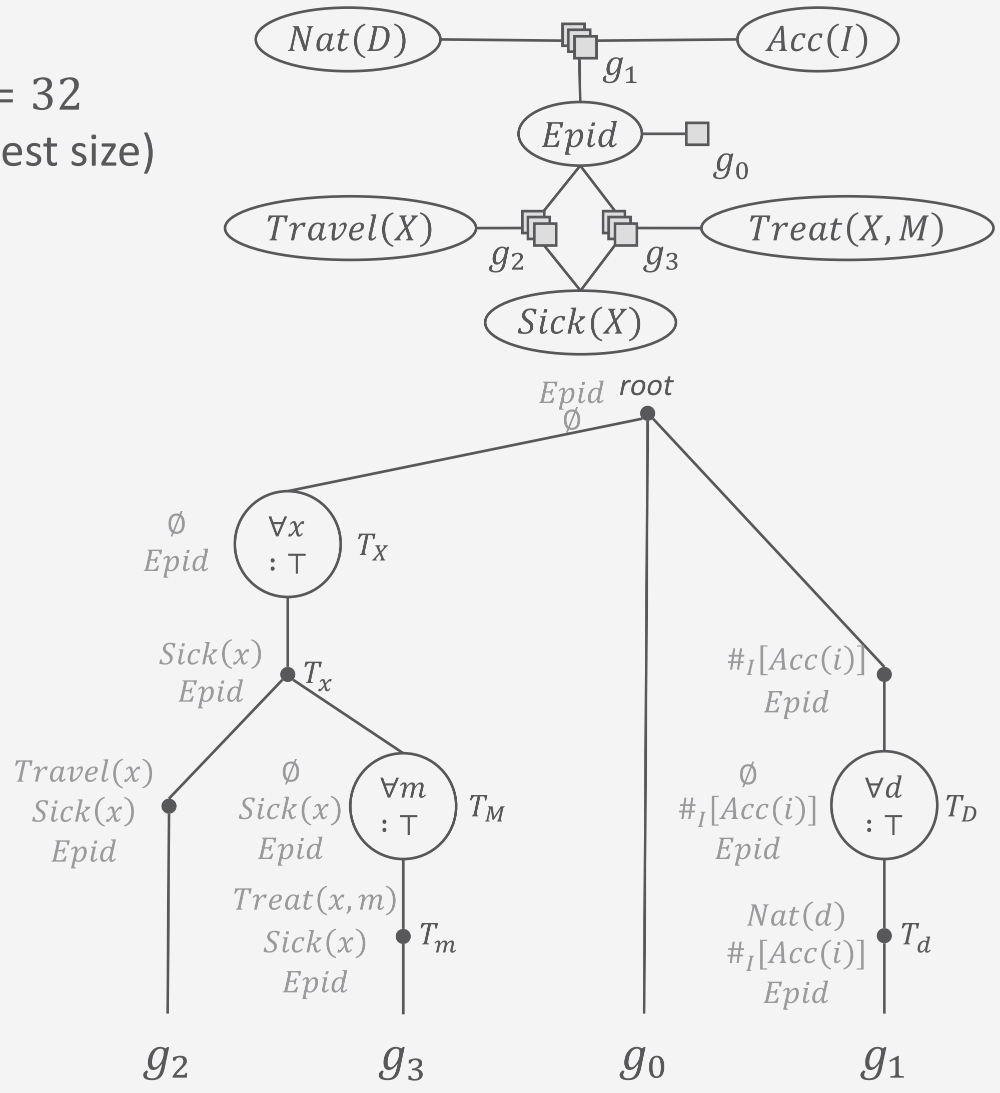
Worst-case Parfactor Size

E.g., with $w_g = 3$, $w_\# = 1$

$$2^3 \cdot |dom(I)|^{2\cdot 1} = 2^3 \cdot 2^2 = 8 \cdot 4 = 32$$

- $32 > 12$ (actual largest size)

- Given lifted width $w_T = (w_g, w_\#)$
- Worst-case parfactor size:
  - Worst case: $w_g + w_\#$ variables in one parfactor
  - Worst-case range size for the w_g PRVs:
    $$r = \max_{A \in rv(G)} |ran(A)|$$
  - Worst-case range size for the $w_\#$ CRVs:
    $$\binom{n_\# + r_\# - 1}{n_\# - 1} \leq {n_\#}^{r_\#}$$
    - $n_\#$ largest domain size of any counted logical variable
    - $r_\#$ largest range size of any of the PRVs in the CRVs
  - Number of mappings by w_g and $w_\#$:
    $$r^{w_g} \cdot ({n_\#}^{r_\#})^{w_\#} = r^{w_g} \cdot {n_\#}^{r_\# w_\#}$$



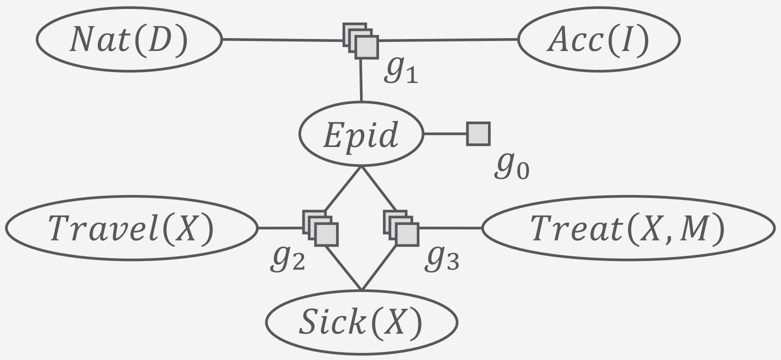
Complexity

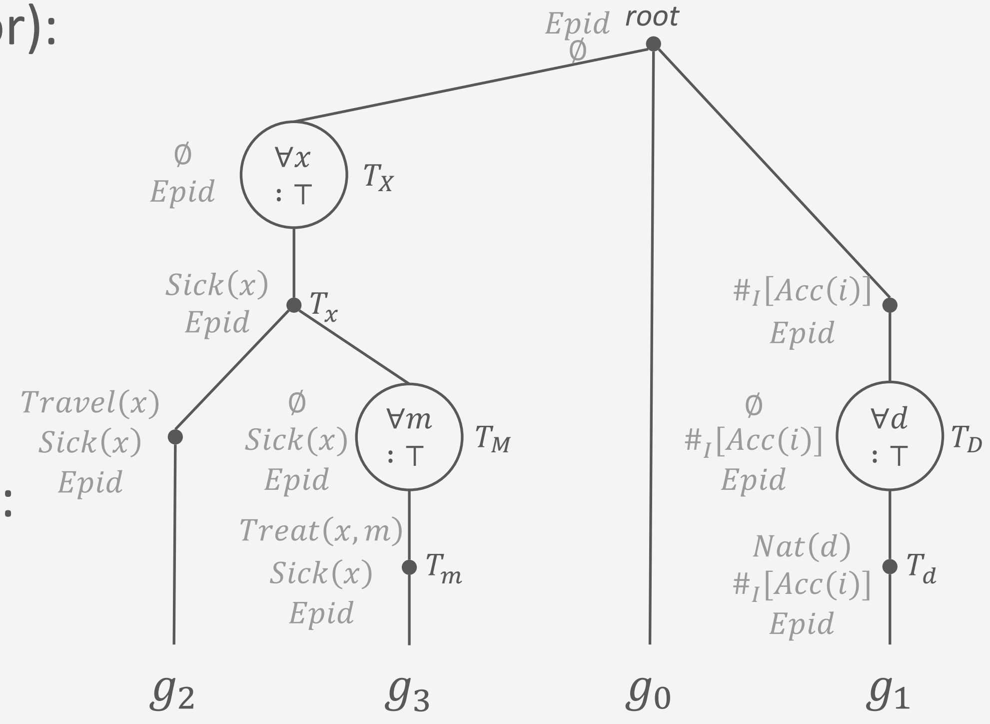
- Worst-case parfactor size $r^{w_g} \cdot {n_\#}^{r_\# w_\#}$
- Complexity of lifted operations
  - Multiplication (goes through each line of each parfactor):
    $O(r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
  - Summation (goes through each line):
    $O(r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
  - Exponentiation (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:
      $O(\log_2(n_\#) \cdot r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
  - Bounded by $O(\log_2(n) \cdot r^{w_g} \cdot {n_\#}^{r_\# w_\#})$

E.g., with $w_g = 3, w_\# = 1$

$r^{w_g} \cdot {n_\#}^{r_\# w_\#} = 32$

- $\log_2(|dom(X)|) \cdot 32$
- $\log_2(|dom(I)|) \cdot 32$





Complexity

E.g., with $w_g = 3, w_\# = 1$

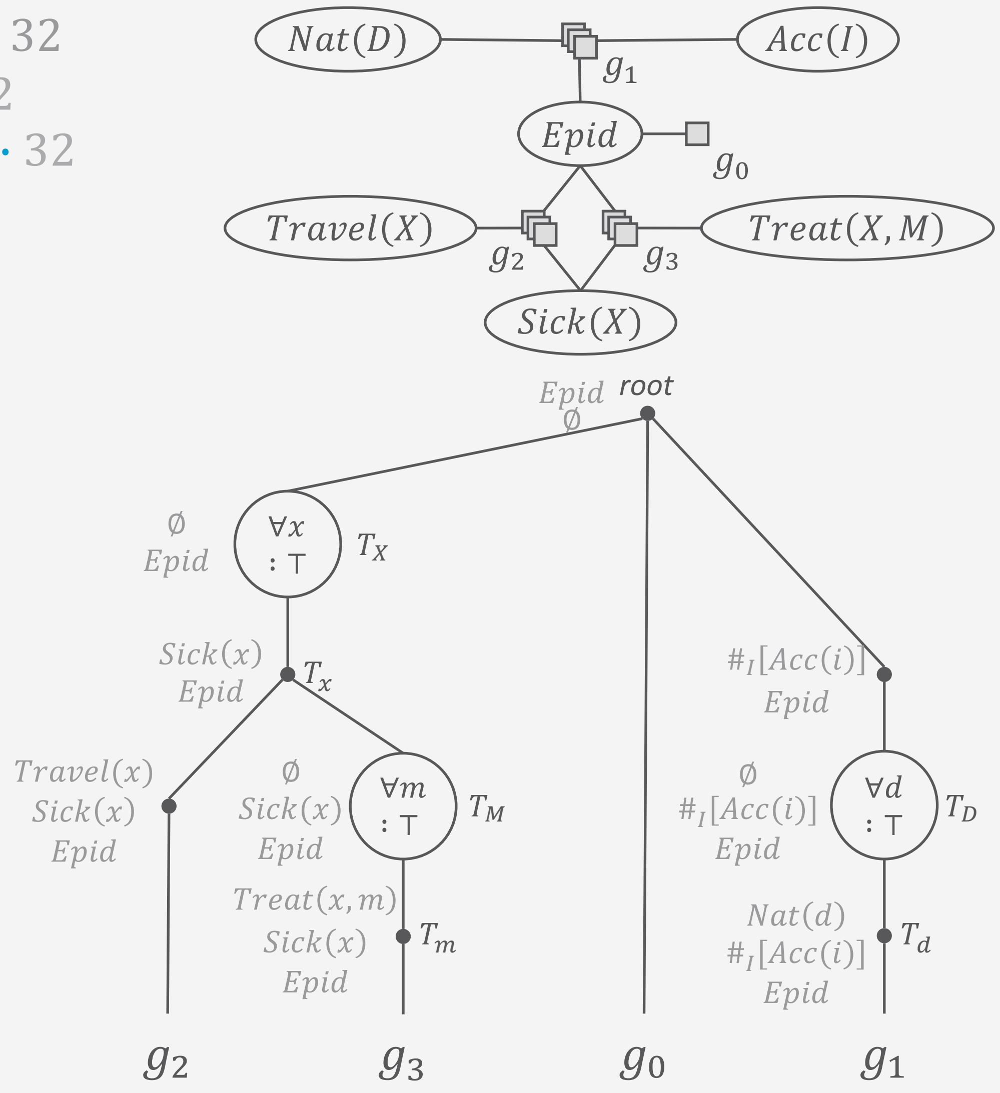
$r^{w_g} \cdot n_\#^{r_\# w_\#} = 32$

- $\log_2(|dom(X)|) \cdot 32$
- $9 \cdot \log_2(|dom(X)|) \cdot 32$

- Worst-case parfactor size $r^{w_g} \cdot n_\#^{r_\# w_\#}$
- Complexity of lifted operations
  - Bounded by $O(\log_2(n) \cdot r^{w_g} \cdot n_\#^{r_\# w_\#})$
- 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 $\max_{g \in G} |\mathrm{rv}(g)|$

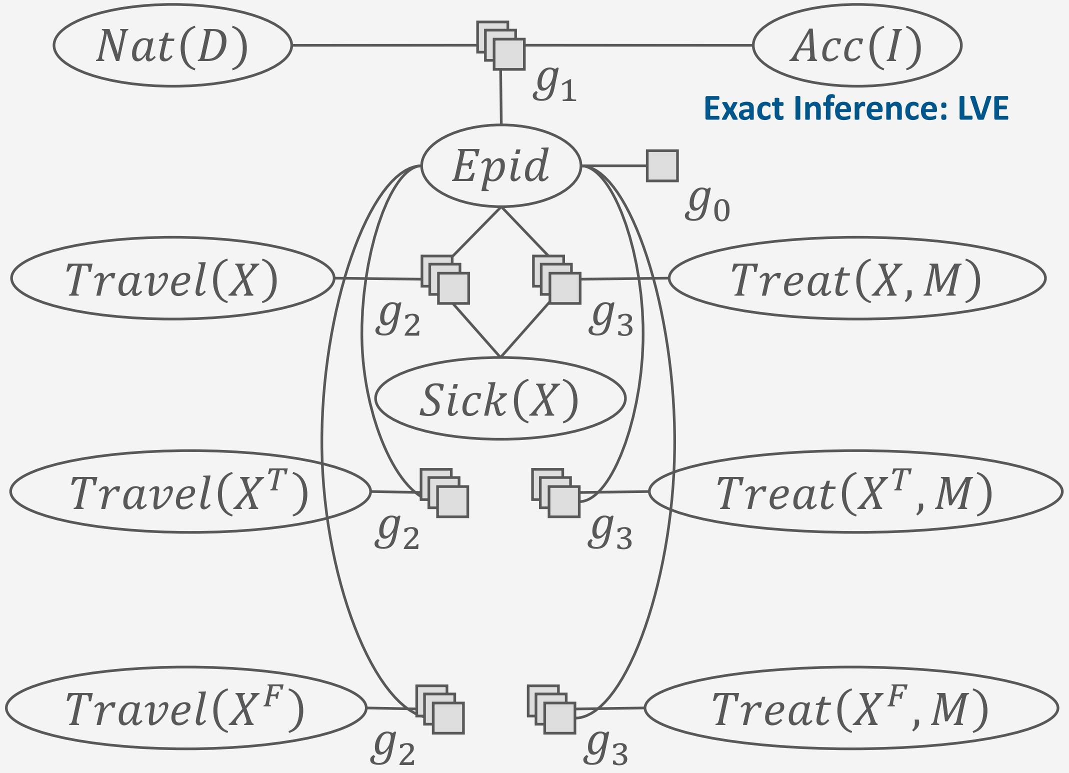


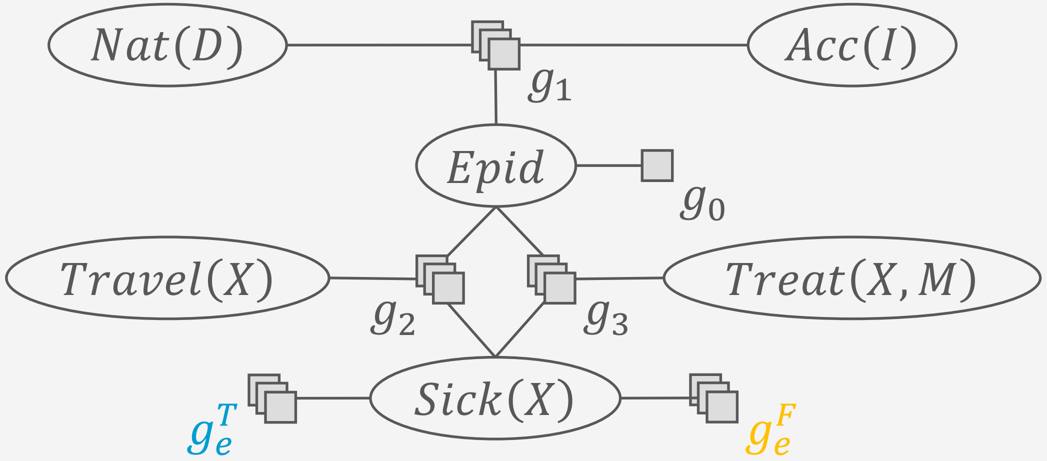
WWU MÜNSTER

Evidence

Exact Inference: LVE

- Absorption complexity: $O(\log_2(n) \cdot r^{w_g} \cdot {n_\#}^{r_\# w_\#})$
  - Collects a subset of lines that still depends exponentially on the largest parfactor size
  - Exponentiates result
- Evidence can yield $|ran(A)|$ 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 more than one logical variable can lead to groundings
- If considering evidence handling as an offline pre-processing step, one could also analyse the model after handling evidence



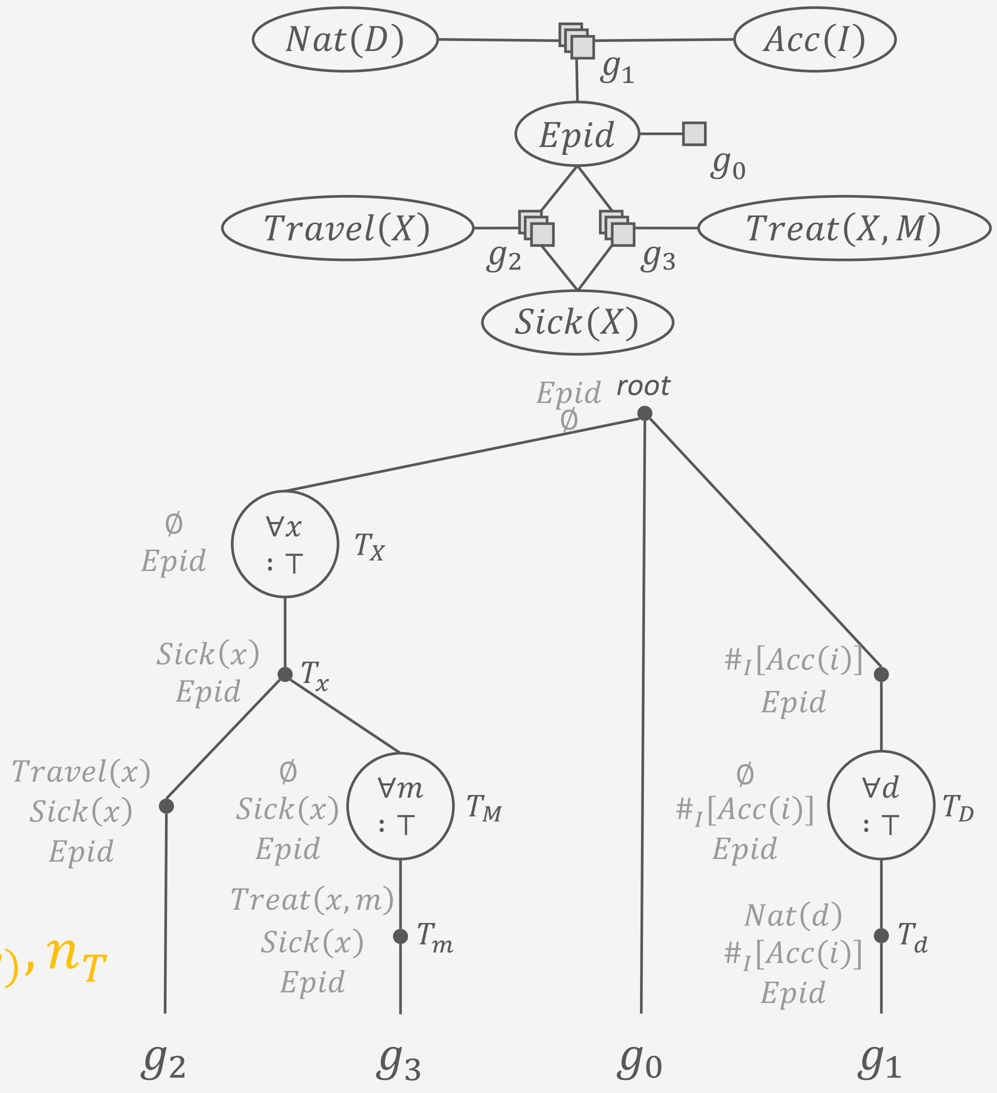


Comparison

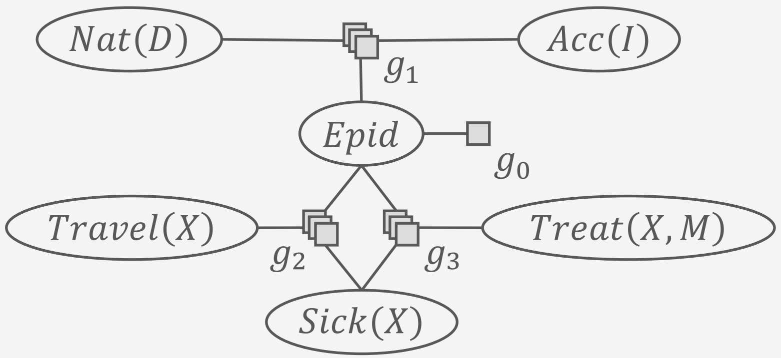
- Complexity of LVE given a liftable, counted FO dtree T for a counted model G:

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

  - $n_T = |rv(G)| + |lv(G)|$
- 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., $w_\# = 0$,
  - $n^{r_\# \cdot 0} = 1 \rightarrow O\left(n_T \cdot \log_2(n) \cdot r^{w_g}\right)$
  - $w = w_g$
  - Difference in $\log_2(n)$ for lifted computations and $n_{gr(T)}, n_T$
    - More noticeable if domain sizes increase ($n_{gr(T)} \gg n_T$)



Comparison



- If count conversions involved, i.e., $w_{\#} > 0$,
  - $w \gg (w_g + w_{\#})$
    - CRV with counted logical variable of domain size n appears grounded in a factor
    - With one count conversion, $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_{\#}})$ vs. $O(n_{gr(T)} \cdot r^{n+c})$
      - c the number of random variables also occurring in the cluster
      - E.g., with $c = 2$:

$$\phi_1(E, Nat(D), Acc(I)) \xrightarrow{\#} \phi_1^{\#}(E, Nat(D), \#_I[Acc(I)]) \xrightarrow{\Sigma} \phi_1(E, \#_I[Acc(I)])$$

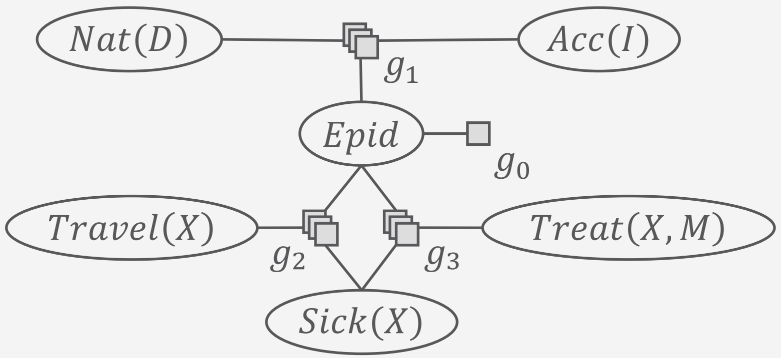
$$\left.\begin{matrix}\phi_1(E, Nat(d_1), Acc(i_1)) \\ \vdots \\ \phi_1(E, Nat(d_1), Acc(i_n))\end{matrix}\right\} \xrightarrow{\cdot} \phi_1^1(E, Nat(d_1), Acc(i_1), \dots, Acc(i_n)) \xrightarrow{\Sigma} \phi_1^1(E, Acc(i_1), \dots, Acc(i_n))$$

$$\vdots$$

$$\left.\begin{matrix}\phi_1(E, Nat(d_m), Acc(i_n)) \\ \vdots \\ \phi_1(E, Nat(d_m), Acc(i_n))\end{matrix}\right\} \xrightarrow{\cdot} \phi_1^1(E, Nat(d_m), Acc(i_1), \dots, Acc(i_n)) \xrightarrow{\Sigma} \phi_1^1(E, Acc(i_1), \dots, Acc(i_n))$$

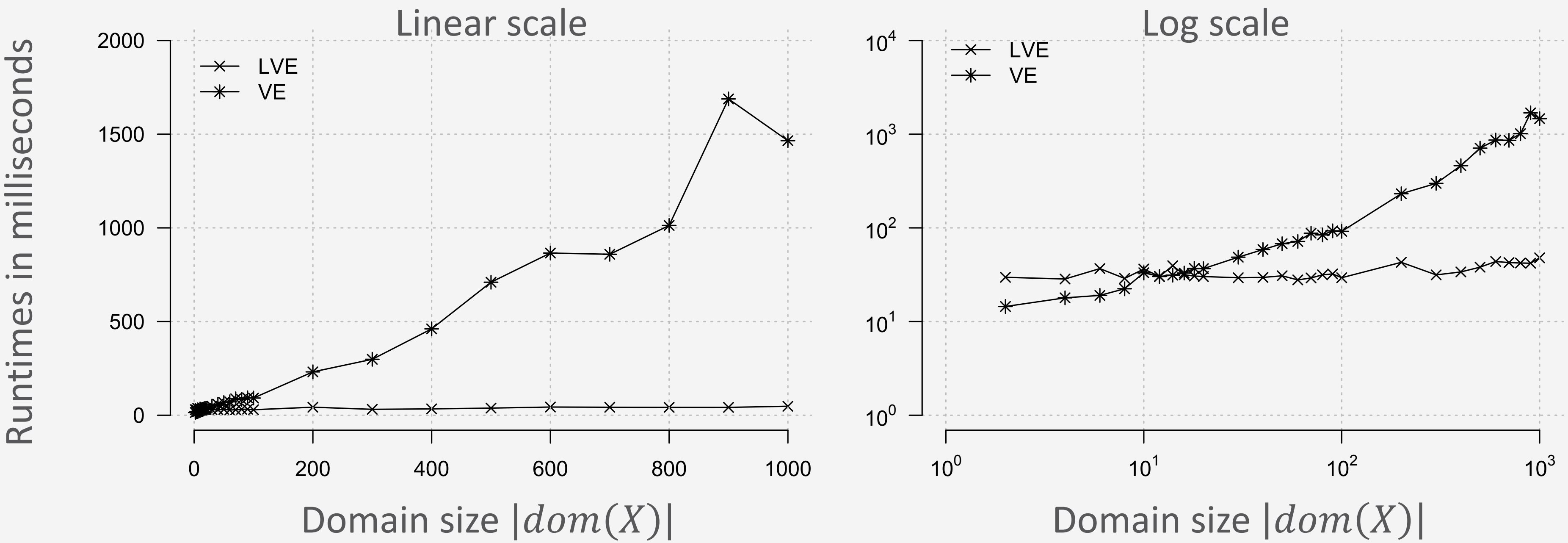
In the lifted case, domain size n no longer occurs in an exponent whereas it does in the propositional case thanks to count conversion

$E = Epid$

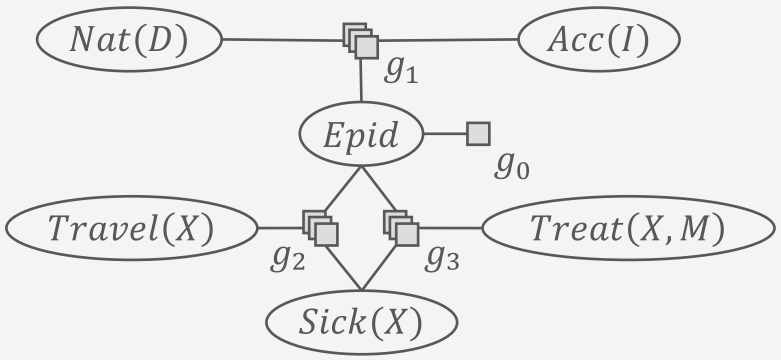


Comparison: Runtime

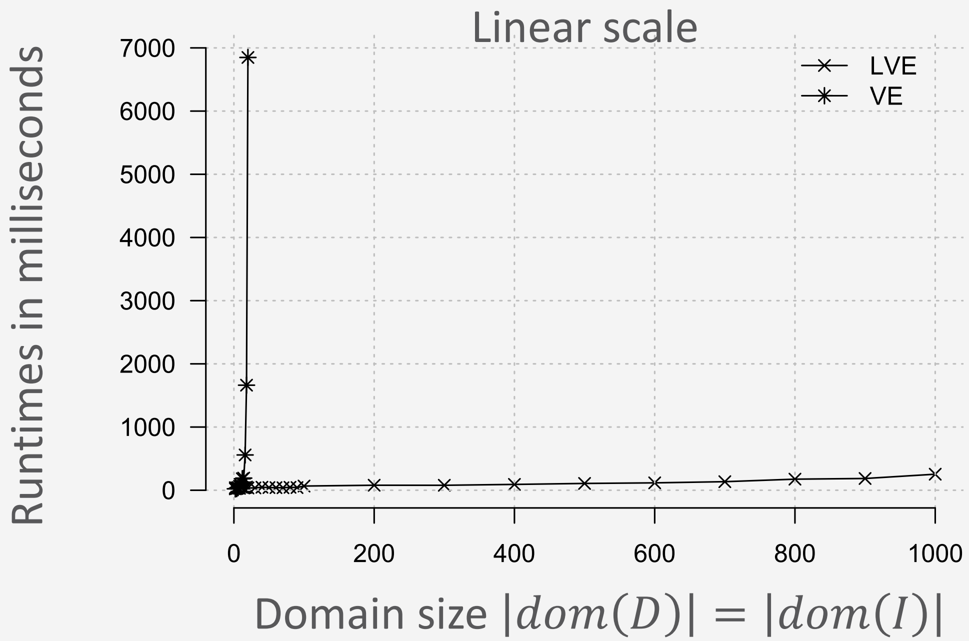
- One count conversion, i.e., $w_\# = 1$,
  - $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\#})$ vs. $O\left(n_{gr(T)} \cdot r^{n+c}\right)$
  - Consider domain size of counted logical variable constant:

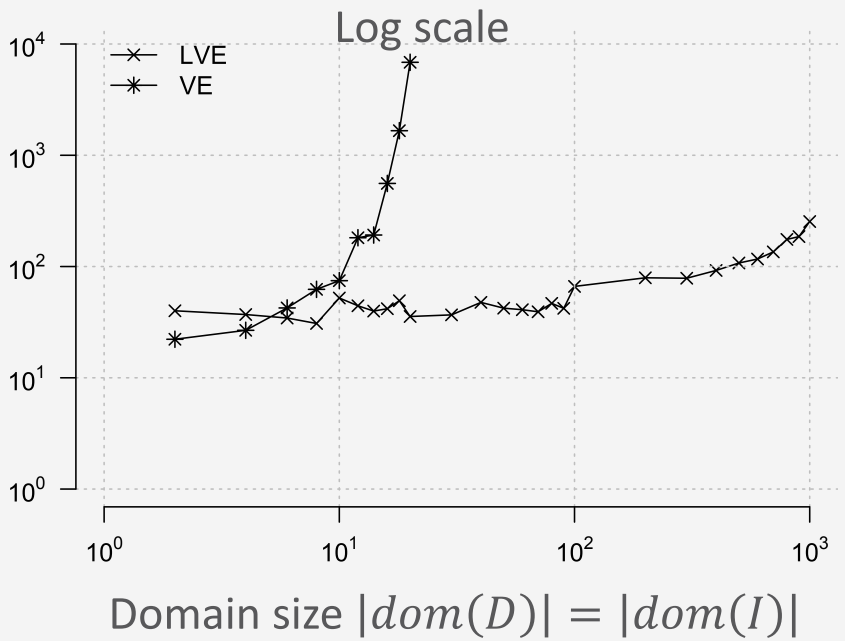


Comparison: Runtime



- 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 I (in g_1) increasing





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: $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$
  - 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
  - $\mathcal{M}^{2lv}$:
    - Two logical variables per parfactor max

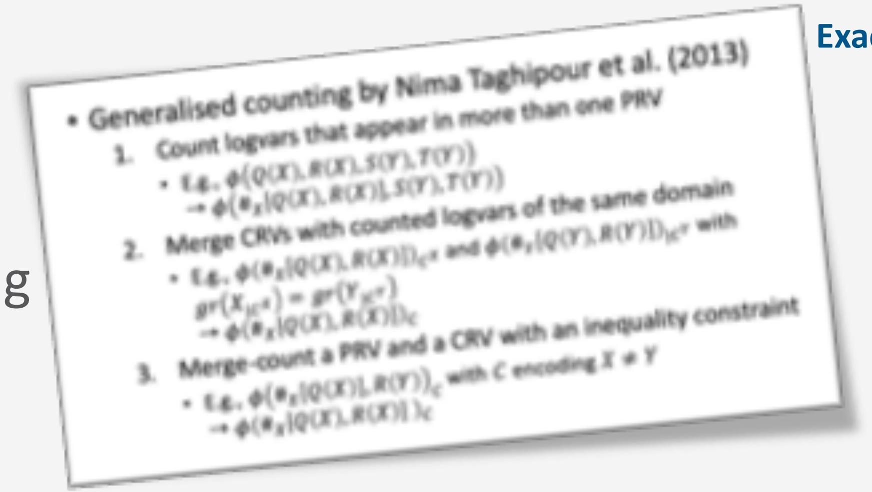
$$g\big(A(X,Y), B(X,Y)\big)$$
$$g(A(X,Y), C(X), C(Y)), X \neq Y$$
$$g(A(X,Y), D(X), E(Y))$$

  - $\mathcal{M}^{1prv}$:
    - One logical variable per PRV (arbitrarily many logical variables per parfactor)

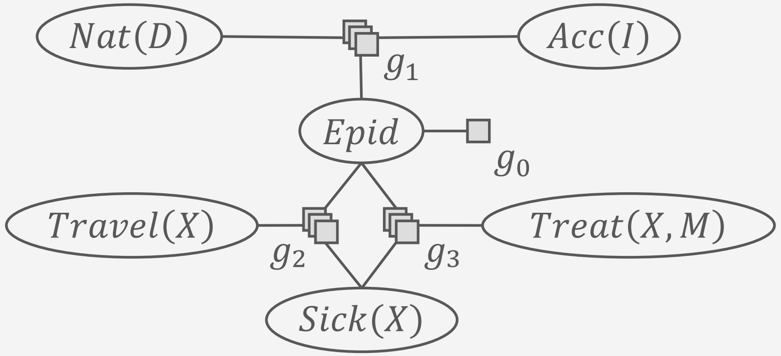
$$g\big(A(X), B(Y), C(Z)\big)$$

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

Completeness



- LVE is complete for $\mathcal{M}^{1prv}$ with generalised counting
  - $\mathcal{M}^{1prv}$: One logical variable per PRV
- Proof:
  - Fact: Only PRVs with one logical variable to eliminate
    1. Perform count conversion on all logical variables in the model; possible scenarios in each parfactor
       A. Logical variable is the only one with a particular domain → Standard count conversion applies
       B. Logical variable occurs in several PRVs without inequality constraints → Generalised Counting 1 applies
       C. Logical variable occurs in several PRVs with inequality constraints → After count-converting PRVs of Scenario B, Generalised Counting 3 applies
      - Afterwards: No uncounted logical variables remain
    2. Multiply all parfactors into one large parfactor and merge CRVs (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 same grounded random variables
    - Sums out PRVs $\{A_1, \dots, A_k\}$ 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 logical variables X_i, X_j, $dom(X_i) = dom(X_j)$

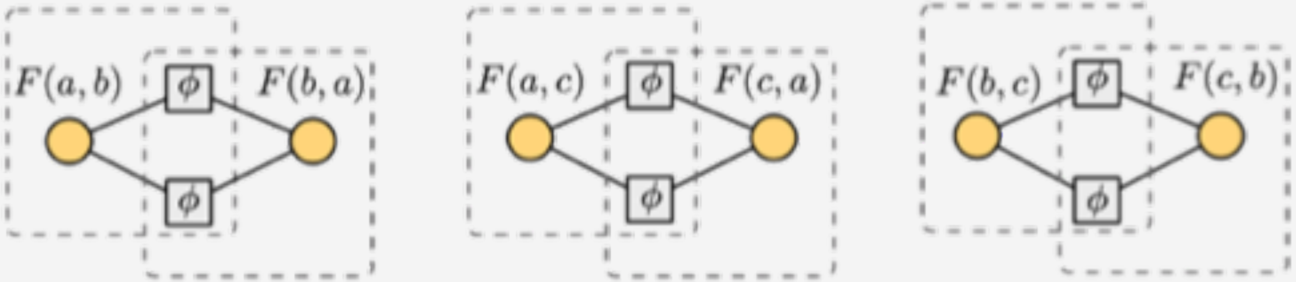
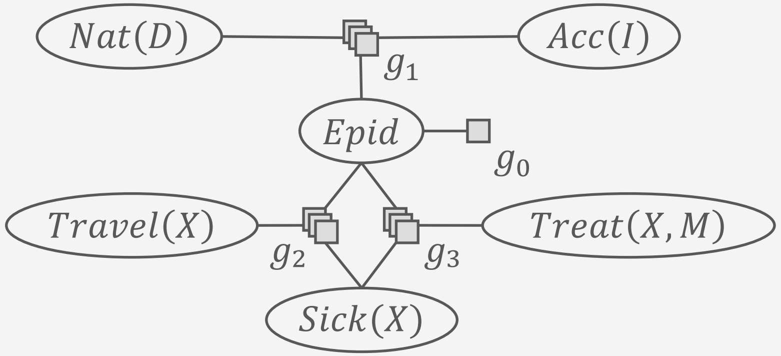


Figure taken from: Nima Taghipour: Lifted Probabilistic Inference by Variable Elimination. *PhD Thesis*, 2013.
Group Inversion: Nima Taghipour, Daan Fierens, Guy Van den Broeck, Jesse Davis, and Hendrik Blockeel: Completeness Results for Lifted Variable Elimination. In: *AISTATS-13 Proceedings of the 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 logical variables X, Y at most
  - Fact 2: Once PRVs with two logical variables are eliminated, model is in $\mathcal{M}^{1prv}$

1. Multiply all parfactors together that share PRVs with two logical variables
   - Preserves the number of logical variables per parfactor, namely, two
2. Eliminate each PRV with two logical variables in each parfactor; possible scenarios
   A. Only PRVs with two logical variables and no inequality constraint → Eliminate using summing out
   B. PRVs with two logical variables with an inequality constraint → Eliminate using group inversion
   - Afterwards: Only PRVs with one logical variable and propositional random variables remain (Fact 2)
3. Count logical variables 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

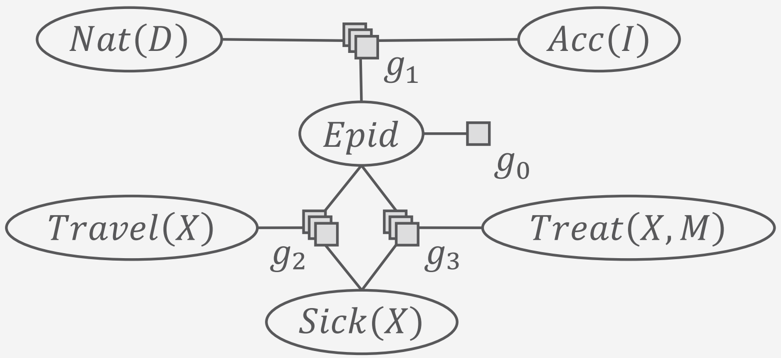
$$g\big(A(X,Y,Z), B(X,Y), C(X)\big) \rightarrow \ liftable$$

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

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

- Completeness results assume a liftable class of queries $\mathcal{Q}$ and a liftable class of evidence $\mathcal{E}$

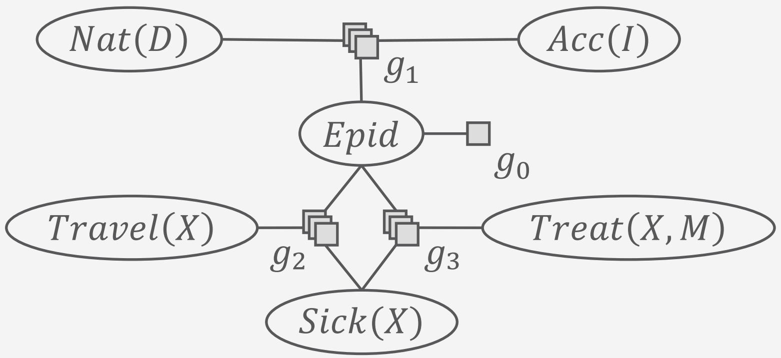
Completeness Beyond Models: Queries



- Queries Q:
  - Class of one ground query term Q *liftable*
    - As argued on earlier slide, one query term does not influence complexity and cannot cause groundings
  - Class of sets of ground query terms $\boldsymbol{Q}$ *not liftable*
    - Proof by counter example
      - $P(Sick(eve), Travel(alice), Treat(bob, m_1))$ grounds X
      - LVE no longer polynomial in domain size
  - Class of query terms $\boldsymbol{Q}$ containing at most one constant for each logical variable in $lv(G)$ *liftable*
    - Argument: Splits do not lead to a set of parfactors whose size depends on the domain size of logical variables
    - Examples: $P(Travel(eve)), P(Travel(eve), Sick(eve)), P(Travel(eve), Nat(chem))$

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.

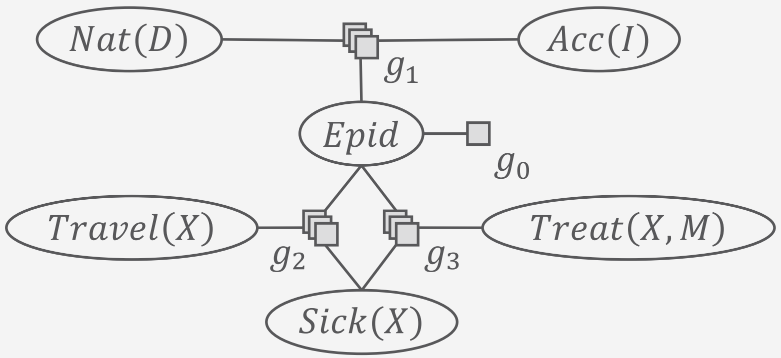
Exact Inference: LVE



Completeness Beyond Models: Queries

- Queries Q:
  - Class of all parameterised queries *not liftable*
    - Proof by counter example, using constraints or logical variables :
      - $P\big(Sick(X'), Travel(X'')\big)_{|\big((X',X''),\{alice,eve\}\times\{eve,bob\}\big)}$
      - Query $P\big(B(X,Y)\big)$ in model $g\big(A(X), B(X,Y), C(Y)\big)$
  - Parameterised query terms with only one parameter per term and one subset of constants per domain *liftable*
    - Proof along the lines of proving completeness for $\mathcal{M}^{1prv}$
    - Example: $P\big(Sick(X), Travel(X)\big)_{|\top}$
- Corollary
  - CRVs compactly represent the result of liftable queries

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.



Completeness Beyond Models: Evidence

- Evidence $\mathcal{E}$:
  - *Liftable* class: Evidence on propositional random variables
    - Example: $Epid = true$
  - *Liftable* class: Evidence on instances of PRVs with one logical variable
    - Example: $Sick(X) = true, dom(X) = \{alice, eve, bob\}$
  - General evidence on PRVs with two logical variables *not 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.

Complexity

- Given liftable query over query terms $\boldsymbol{Q}$
  - Class of query terms $\boldsymbol{Q}$ containing at most one constant for each logical variable in $lv(G)$ if ground or one set of constants if parameterised
- Assumption is 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 $O(n_T \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$
- Liftable parameterised queries require only at most q additional count conversions, which are bounded by $O(\log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#})$, and hopefully, $q \ll n_T$

- I.e., LVE complexity given a liftable model, a liftable query, and liftable evidence remains at

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

Tanya B. Rescued from a Sea of Queries: Exact Inference in Probabilistic Relational Models. PhD Thesis, 2020.

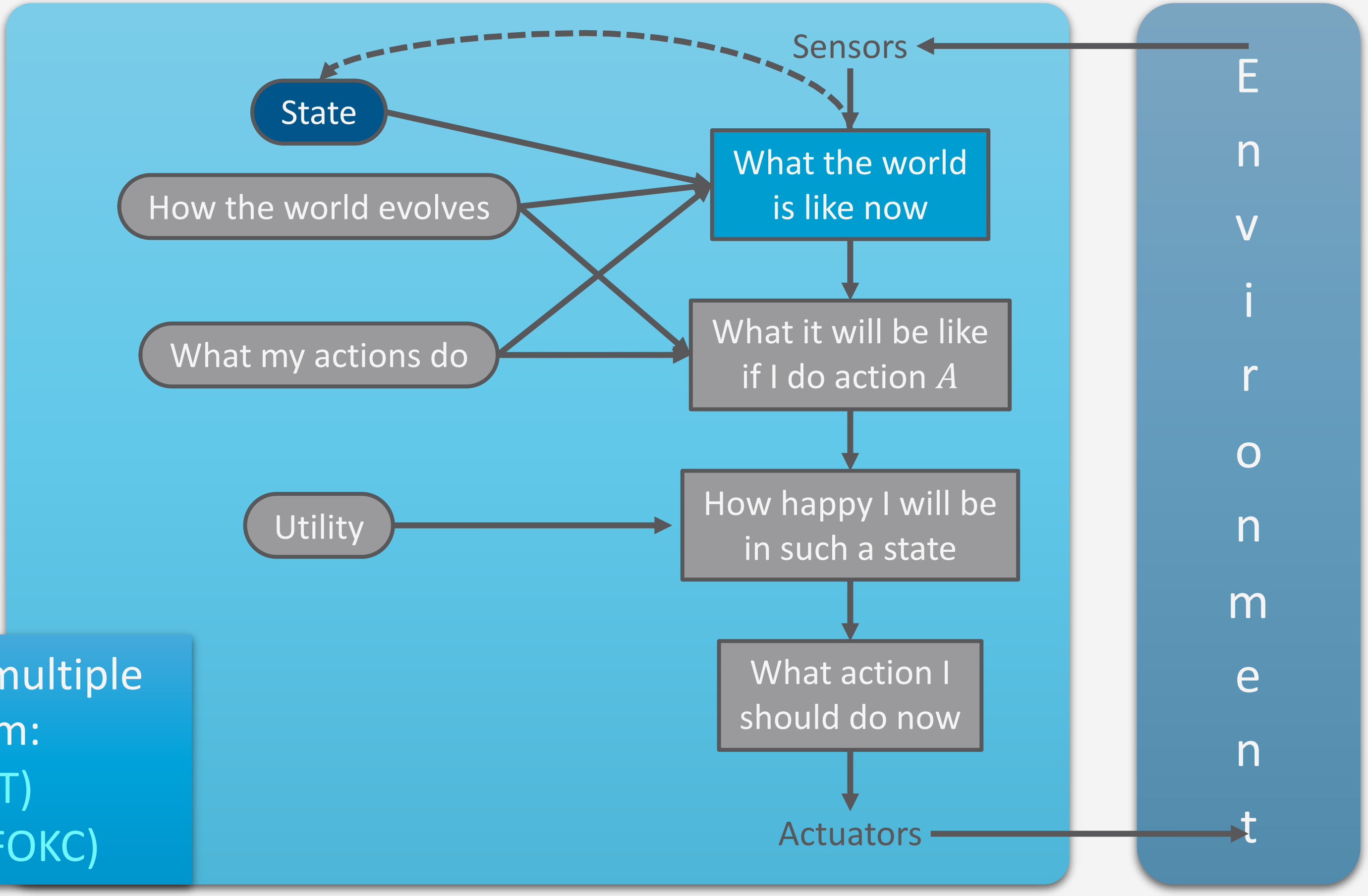
Interim Summary

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

Contents in this Lecture Related to *Utility-based Agents*

- Further topics
  3. (Episodic) PRMs
  4. Lifted inference (in episodic PRMs)
  5. Lifted learning (of episodic PRMs)
  6. Lifted sequential PRMs and inference
  7. Lifted decision making
  8. Continuous space and lifting

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)

Outline: 4. Lifted Inference

A. Exact Inference

i. Lifted Variable Elimination for Parfactor Models
   - Idea, operators, algorithm, complexity

ii. Lifted Junction Tree Algorithm
   - Idea, helper structure: junction tree, algorithm

iii. First-order Knowledge Compilation for MLNs
   - Idea, helper structure: circuit, algorithm

B. Approximate Inference: Sampling

- Rejection sampling
- (Lifted) likelihood sampling
- (Lifted) Markov Chain Monte Carlo sampling

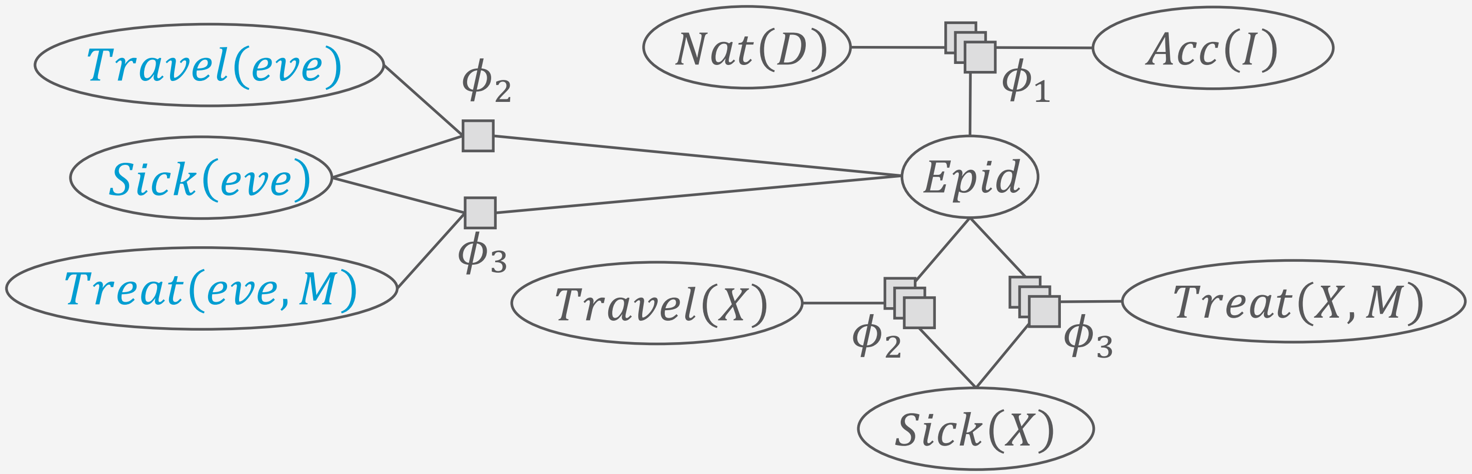
Appendix

Example Calculation with a Greedy Size-based Heuristics

Example Model without g_0

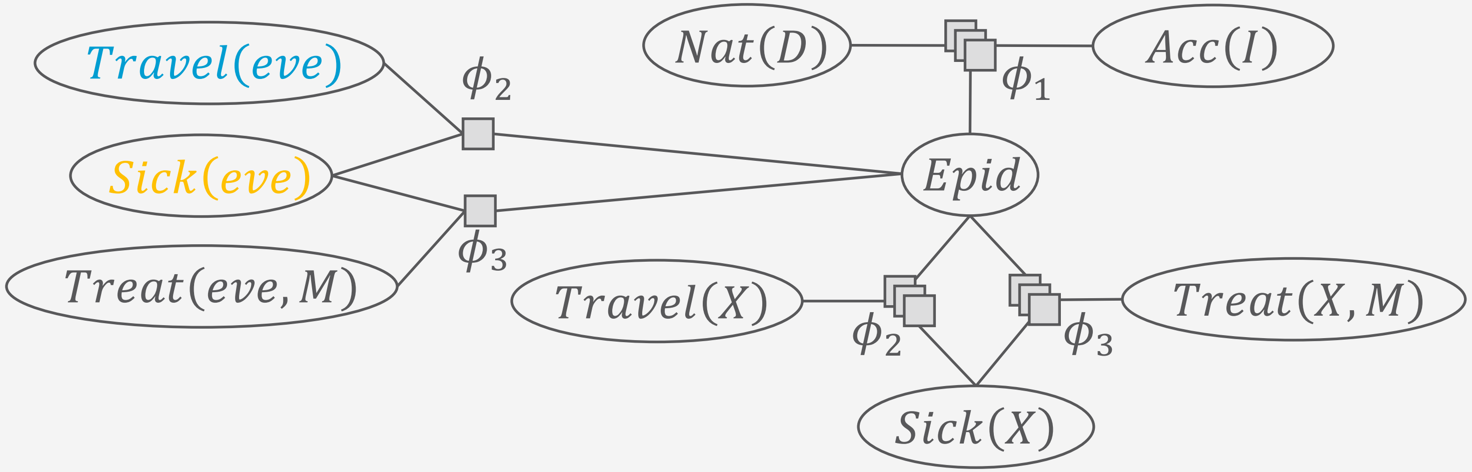
LVE: Example

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



LVE: Example

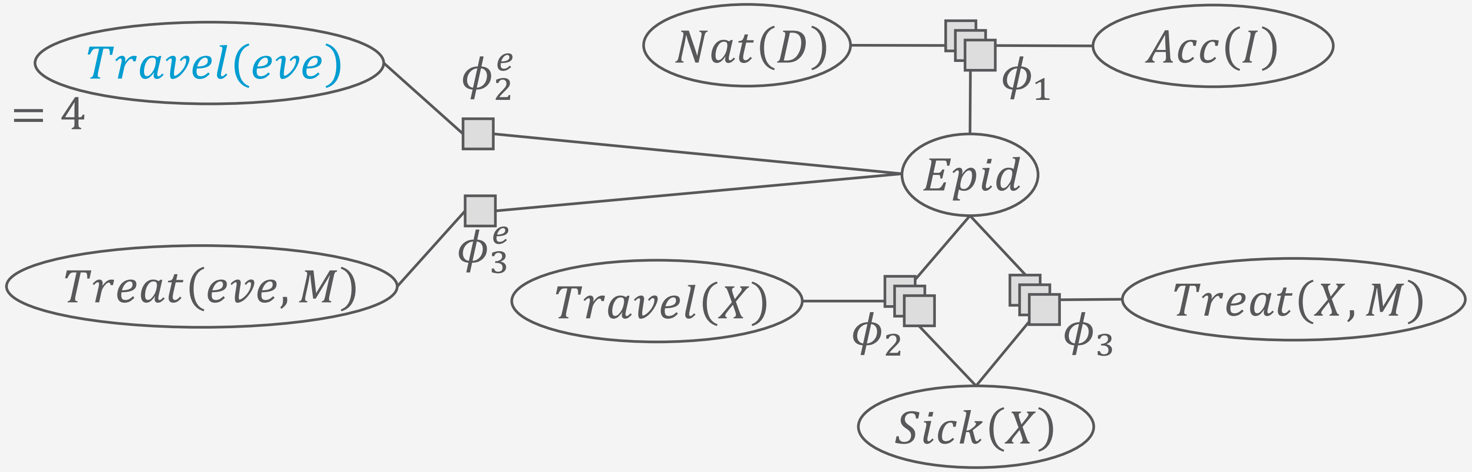
- Absorbing evidence $Sick(eve) = true$:
  - Absorb evidence in $\phi_2(Epid, Sick(eve), Travel(eve))$
    - Yields $\phi_2^e(Epid, Travel(eve))$
  - Absorb evidence in $\phi_3(Epid, Sick(eve), Treat(eve, M))$
    - Yields $\phi_3^e(Epid, Treat(eve, M))$



LVE: Example

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

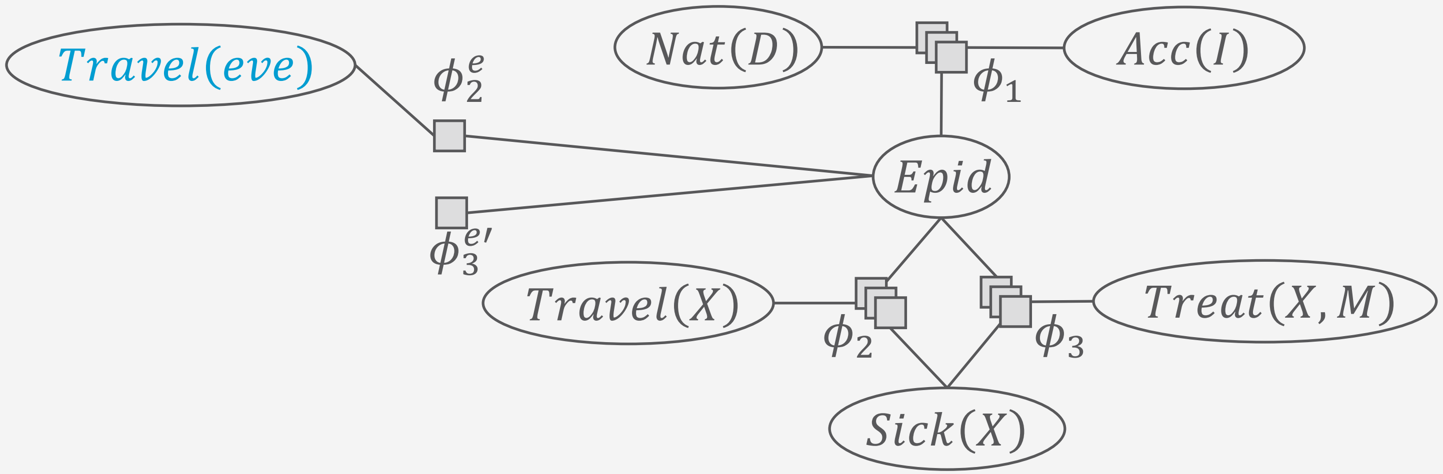
Smallest size



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: $2^2 = 4$

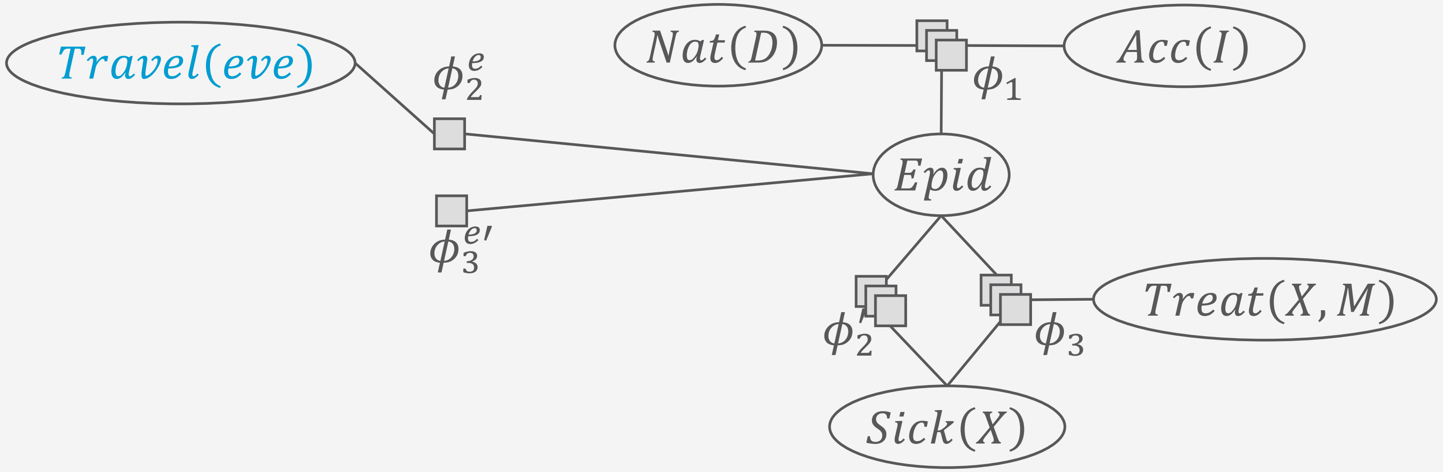
Chosen at random



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

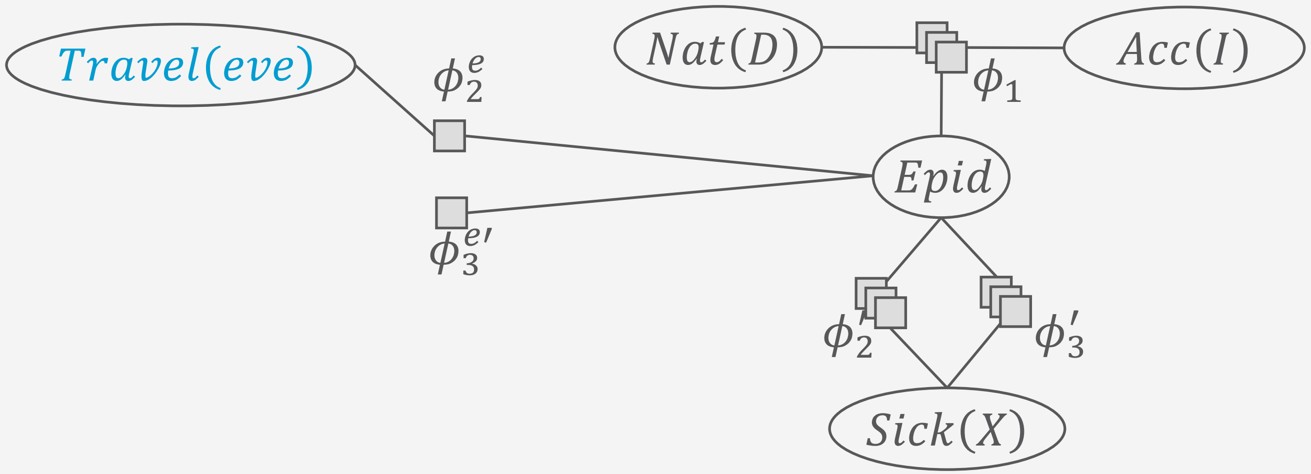


LVE: Example

- 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: $2^2 = 4$

Chosen at random

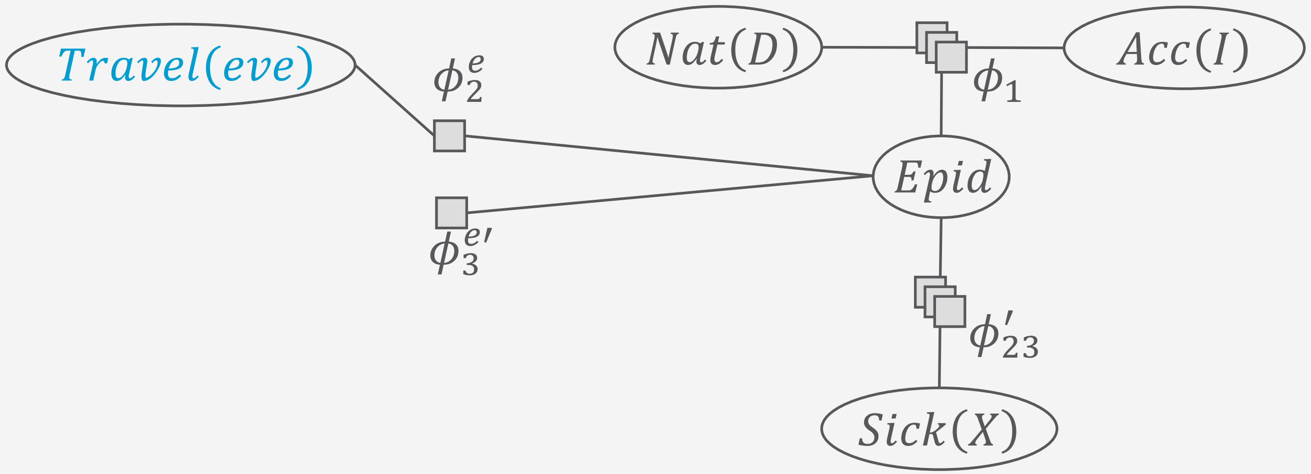
    - Multiply ϕ^e_2 and $\phi^{e\prime}_3$
      - Yields $\phi'_{23}(Epid, Travel(eve)) \rightarrow$ size: $2^2 = 4$
    - Count-convert $Nat(D)$
      - Yields $\phi^D_1(Epid, \#_D[Nat(D)], Acc(I))$ $\rightarrow$ size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi^I_1(Epid, Nat(D), \#_I[Acc(I)])$ $\rightarrow$ size: $2^2 \cdot 3 = 12$



LVE: Example

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

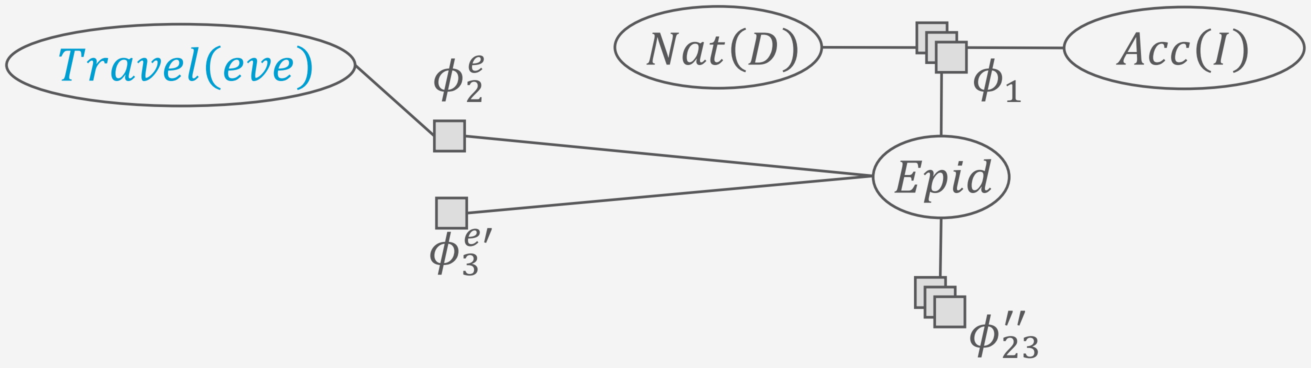
Only one



LVE: Example

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

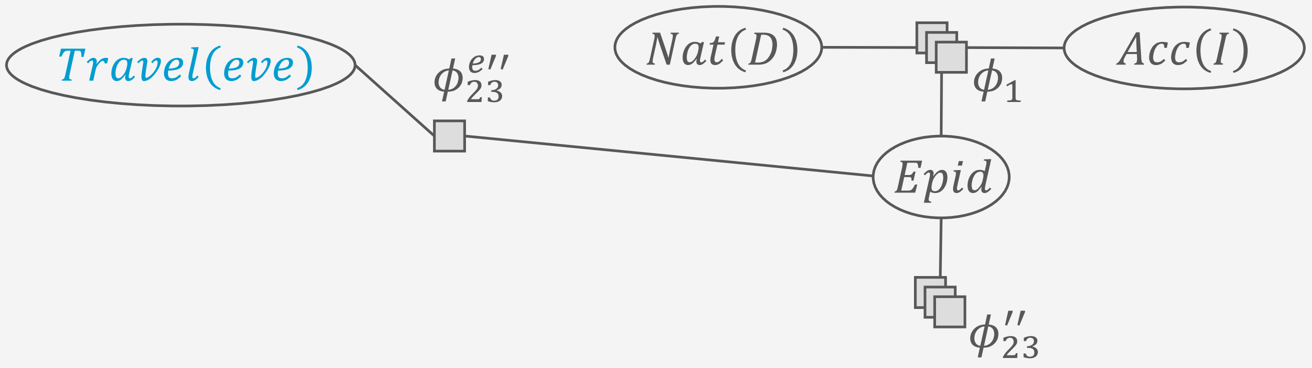
Smallest size



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 $\phi_1^D(Epid, \#_D[Nat(D)], Acc(I))$ → size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Man(W)$
      - Yields $\phi_1^I(Epid, Nat(D), \#_I[Acc(I)])$ → size: $2^2 \cdot 3 = 12$

Smallest size



LVE: Example

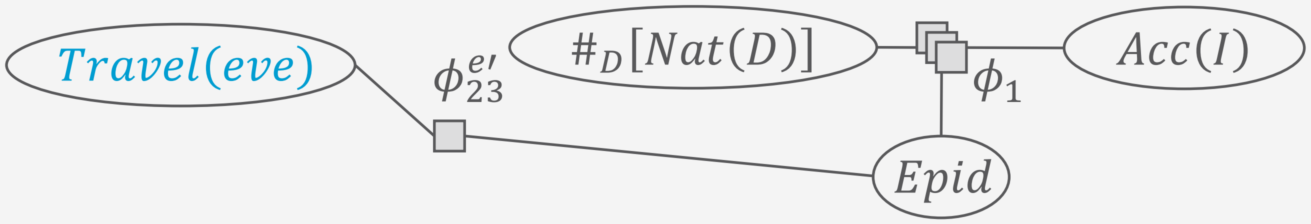
- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Count-convert $Nat(D)$
      - Yields $\phi_1^D(Epid, \#_D[Nat(D)], Man(W))$ → size: $2 \cdot 3 \cdot 2 = 12$
    - Count-convert $Acc(I)$
      - Yields $\phi_1^I(Epid, Nat(D), \#_I[Acc(I)])$ → size: $2^2 \cdot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

Chosen at random

LVE: Example

- Eliminate all non-query terms
  - PRVs fulfilling sum-out preconditions:
    - $Acc(I)$
      - Yields $\phi'_1(Epid, \#_D[Nat(D)])$ → size: $2 \cdot 3 = 6$

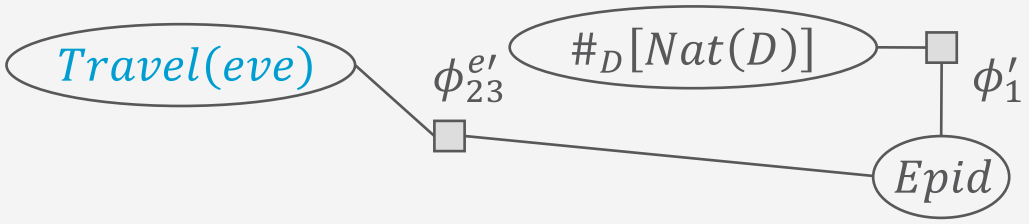
Only one



LVE: Example

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

Only one



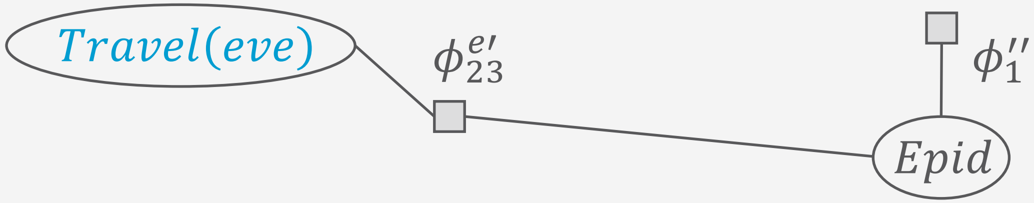
No uncounted logvars left (basically standard VE + CRVs)

LVE: Example

- Eliminate all non-query terms
  - No PRVs fulfilling sum−out preconditions; others:
    - Multiply ϕ''_1 and $\phi^{e\prime}_{23}$
      - Yields $\phi^{e\prime\prime}_{123}(Travel(eve), Epid) \rightarrow$ size: 4

Only one

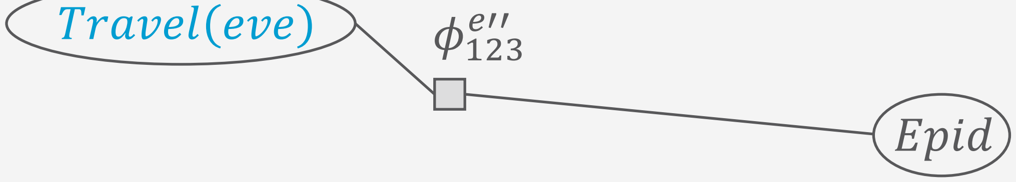
Only propositional and ground random variables left (standard VE)



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'

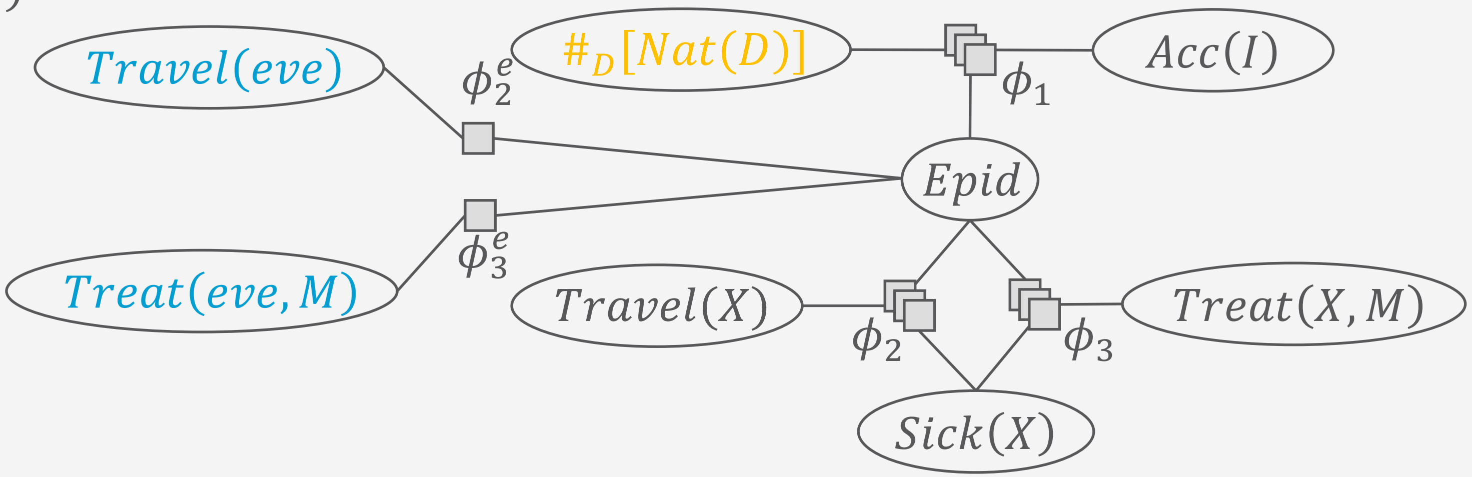
LVE: Example – Complete Derivation

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

$$= \frac{1}{Z} \sum_{e \in ran(Epid)} \phi_2^e(Travel(eve), e) \sum_{h_n \in ran(\#_D[Nat(D)])} Mul(h_n) \left(\sum_{a \in ran(Acc(I))} \phi_1(e, h_n, a) \right)^{|dom(I)|}$$

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

$$\sum_{\substack{te \in \\ ran(Treat(eve,M))}} \phi_3(e, te)$$



- After shattering, absorption, and the required count conversion