Intelligent Agents:
Web-mining Agents

Probabilistic Graphical Models

Lifted Inference

Tanya Braun

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Probabilistic Graphical Models (PGMs)

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

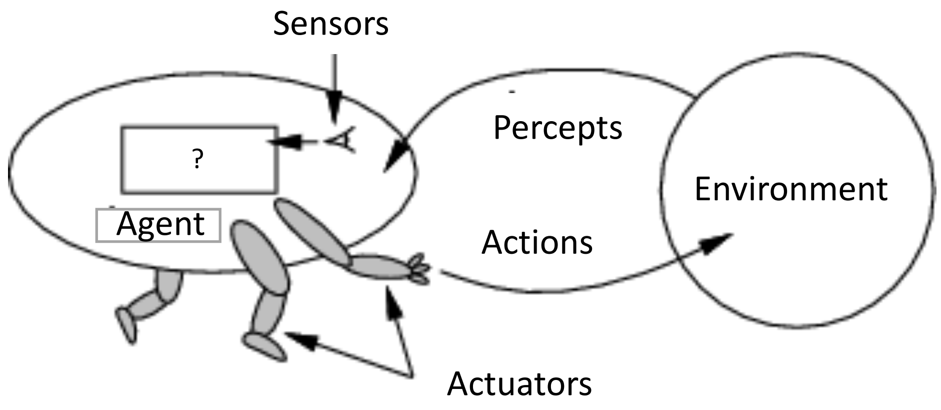
Outline: 3. Lifted Inference

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

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

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

D. ***Beyond Standard Query Answering***
   - Adaptive inference
   - Changing and unknown domains
   - Assignment queries

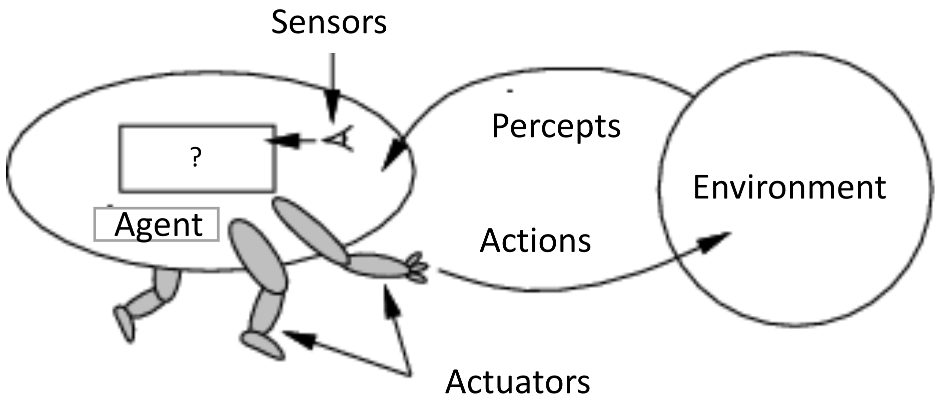


Adaptive Inference

What if one of the inputs changes?

Adaptive Inference

- What if there is a change in the environment?
  - Could be from actions of the agent or from the environment within, e.g., through other agents
    - Changes in percepts → changes in evidence
    - Fundamental changes in environment → changes in model
  - If an agent uses an algorithm with a helper structure given a model G representing the environment and the environment changes, then G has to change as well as the helper structure



Inferring model changes directly using percepts based on an expected increase in utility if assuming a change (or a similar optimisation criterion) is the ultimate goal but we and the science are not really there yet

Adaptive Inference

- Avoid starting from scratch to fast reach the point of answering queries again
  → adaptive inference
- In case of LJT: changes to consider
  - In evidence
  - In model
    - In domain sizes
      - Nice property of relational models:
        Changes in domain sizes do not affect the model structure
        - Propositional models: number of randvars changes, which changes the helper structure
    - In potentials
    - In parfactors (addition, deletion)

Adaptive Evidence Entering

- If evidence changes
  - Standard way: restart at Step 2 (Evidence Entering)
    - Delete existing evidence (if still stored)
    - Reset local models
    - Enter evidence anew
    - Pass messages
    - Answer queries
- If evidence changes only partially or only a subset of nodes is concerned, then
  - Reset of all local models not necessary
- Reset only those local models that are affected by evidence changes $\Delta_E = \{g_e^+ \text{or } g_e^-\}_{e=1}^m$:

  $$\forall \boldsymbol{C}_i \in J : \text{reset } G_i \text{ iff } \exists\, g_e \in \Delta_E : rv(g_E) \subseteq \boldsymbol{C}_i$$

  - g_e^+ new evidence to add, g_e^- old evidence to remove
- if G_i reset, enter evidence anew $\boldsymbol{C}_i$

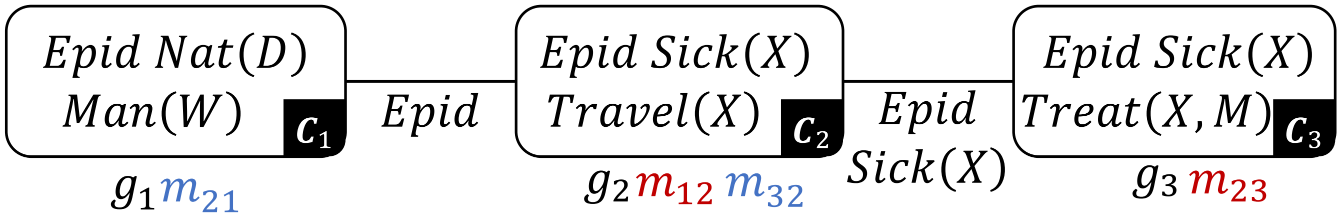
AdaptEvidence

Adaptive Message Passing

- If only some local models change,
  - Calculating all messages anew not necessary
- Message passing under adaptive inference:
  - Pass messages as before based on the two message pass conditions
  - Send empty message $m_{ij} = \emptyset$ to indicate no changes
    - If receiving an empty message, mark message as valid
    - If receiving a not empty message, mark message as changed
  - Calculate a new message m_{ij} if
    - G_i marked changed or
    - At least one message $m_{ki}, k \neq j, k \in nbs(i)$, marked

AdaptMessages

  - E.g.,
    - New evidence on $Nat(D)$ → G_1 changes (absorbs evidence)
    - m_{21}, m_{32} still valid (no changes in $\boldsymbol{C}_2$ or $\boldsymbol{C}_3$)
    - m_{12}, m_{23} need to be updated as they were based on the original G_1



Changes in Model

- Changes in potentials of parfactors g
  - Only change local models but not structure of FO jtree
  - Mark local model where the change of g occurred
  - Use adaptive message passing to update messages
- Changes in domain sizes of logvars X
  - Only change message calculation but not structure of FO jtree
  - Mark local models of parclusters $\boldsymbol{C}_i$ where $X \in lv(\boldsymbol{C}_i)$
  - Use adaptive message passing to update messages
- More complicated:
  Adding or deleting parfactors
  - Replacing a parfactor = Add the new, delete the old one

Adding a Parfactor

- New parfactor g^+ added to model G with FO jtree $J = (V, E)$
  - $\boldsymbol{A}^{old} = rv(g^+) \cap rv(G)$: known PRVs
  - $\boldsymbol{A}^{new} = rv(g^+) \setminus rv(G)$: new PRVs
- Even after adding g^+, J has to fulfil the FO jtree properties, i.e., after adding g^+
  - Property 2 holds: $\exists \boldsymbol{C}_i \in V : rv(g^+) \subseteq \boldsymbol{C}_i$
  - Running intersection property (3) still holds
    - Automatically holds for $\boldsymbol{A}^{new}$ as $\boldsymbol{A}^{new}$ becomes part of exactly one parcluster and does not occur in any other as $\boldsymbol{A}^{new}$ contains only new PRVs
    - For all PRVs in $\boldsymbol{A}^{old}$, the property holds before addition and has to holds afterwards as well
- Goals for addition procedure:
  1. Maintain FO jtree properties
  2. Keep parclusters as small as possible

1. $\forall \boldsymbol{C} \in V : \boldsymbol{C} \subseteq rv(G)$
2. $\forall g \in G : \exists \boldsymbol{C} \in V : rv(g) \subseteq \boldsymbol{C}$
3. If $\exists A \in rv(G) : A \in \boldsymbol{C}_i \wedge A \in \boldsymbol{C}_j$ with $\boldsymbol{C}_i, \boldsymbol{C}_j \in V$, then $\forall \boldsymbol{C}_k \in V$ on the path between $\boldsymbol{C}_i, \boldsymbol{C}_j : A \in \boldsymbol{C}_k$

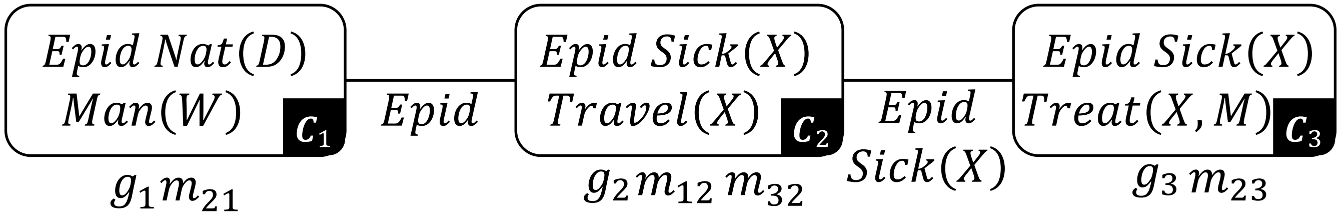
Adding a Parfactor

- If $\boldsymbol{A}^{old}$ occurs in one parcluster $\boldsymbol{C}_i$, easy (ok, easier)
  - $\boldsymbol{A}^{old}$ already fulfils running intersection property
    - Decide what to do based on the existence of $\boldsymbol{A}^{new}$ and how $\boldsymbol{A}^{old}$ relates to $\boldsymbol{C}_i$
  1. If $rv(g^+) \subseteq \boldsymbol{C}_i$: add g^+ to G_i
     - Means that $\boldsymbol{A}^{new} = \emptyset$
  2. If $rv(g^+) \supset \boldsymbol{C}_i$: extend $\boldsymbol{C}_i$ with $\boldsymbol{A}^{new}$, add g^+ to G_i
     - Means that $\boldsymbol{A}^{old} = \boldsymbol{C}_i$
  3. If $\boldsymbol{A}^{old} \subset \boldsymbol{C}_i \wedge \boldsymbol{A}^{new} \cap \boldsymbol{C}_i = \emptyset$, build $\boldsymbol{C}_k = rv(g^+)$ as neighbour to $\boldsymbol{C}_i$, add g^+ to G_k
     - Means $\boldsymbol{S}_{ik} = \boldsymbol{A}^{old}$
     - Means that there are some PRVs in $\boldsymbol{C}_i$ that do not occur in g^+
     - Adding $rv(g^+)$ to $\boldsymbol{C}_i$, i.e., extending $\boldsymbol{C}_i$ with $\boldsymbol{A}^{new}$, unnecessarily enlarges $\boldsymbol{C}_i$

Adding a Parfactor: Example

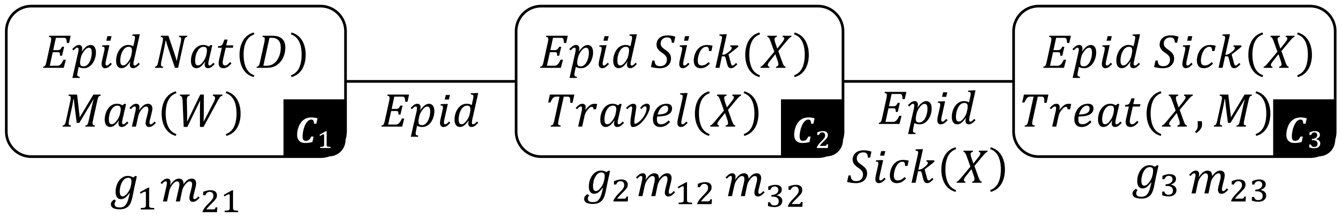
1. If $rv(g^+) \subseteq \boldsymbol{C}_i$: add g^+ to G_i
2. If $rv(g^+) \supset \boldsymbol{C}_i$: extend $\boldsymbol{C}_i$ with $\boldsymbol{A}^{new}$, add g^+ to G_i
3. If $\boldsymbol{A}^{old} \subset \boldsymbol{C}_i \wedge \boldsymbol{A}^{new} \cap \boldsymbol{C}_i = \emptyset$, build $\boldsymbol{C}_k = rv(g^+)$ as neighbour to $\boldsymbol{C}_i$, add g^+ to G_k

- E.g.,
    1. Add $g^+ = \phi(Epid) \rightarrow$ add to any parcluster
    2. Add $g^+ = \phi(Epid, Nat(D), Man(W), Season) \rightarrow$ extend $\boldsymbol{C}_1$ with PRV $Season$, add g^+ to G_1
    3. Add $g^+ = \phi\big(Epid, Sick(X), Work(X)\big) \rightarrow$ build $\boldsymbol{C}_4 = \{Epid, Sick(X), Work(X)\}$ with $G_4 = \{g^+\}$, add as neighbour to either $\boldsymbol{C}_2$ or $\boldsymbol{C}_3$
        - If extending $\boldsymbol{C}_2$ or $\boldsymbol{C}_3$ with PRV $Work(X)$, then largest parcluster size increased to 4 instead of keeping it at 3 for faster query answering



Adjusting FO Jtrees

- What if $\boldsymbol{A}^{old}$ does not occur in one parcluster?
  - E.g., $g^+ = \phi\big(Treat(X,M), Man(W)\big)$
- Cannot simply extend one parcluster or add a new parcluster for g^+
  - Most likely violates the running intersection property
  - E.g.,
    - Cannot extend $\boldsymbol{C}_1$ for $g^+ \rightarrow$ violates running intersection property for $Treat(X,M)$ occurring in $\boldsymbol{C}_3$ but not $\boldsymbol{C}_2$
    - Same holds for extending $\boldsymbol{C}_3$ for g^+ and $Man(W)$
- Adjust FO jtree such that $\boldsymbol{A}^{old}$ occurs in one parcluster maintaining the running intersection property

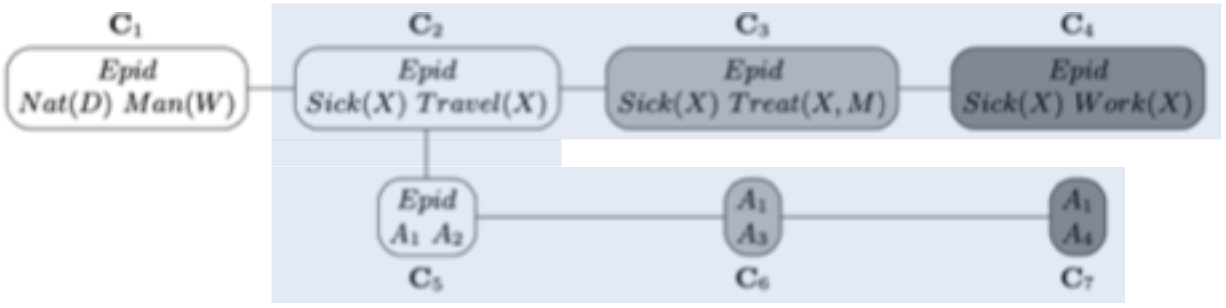


Adjusting FO Jtrees

- Adjust FO jtree such that $\boldsymbol{A}^{old}$ occurs in one parcluster maintaining the running intersection property
  - Find a set of parclusters N such that $\boldsymbol{A}^{old} \subseteq rv(N)$
  - Adjust FO jtree in way that afterwards $\exists \boldsymbol{C}_i : \boldsymbol{A}^{old} \subseteq \boldsymbol{C}_i$ using an adjustment strategy

Adjust

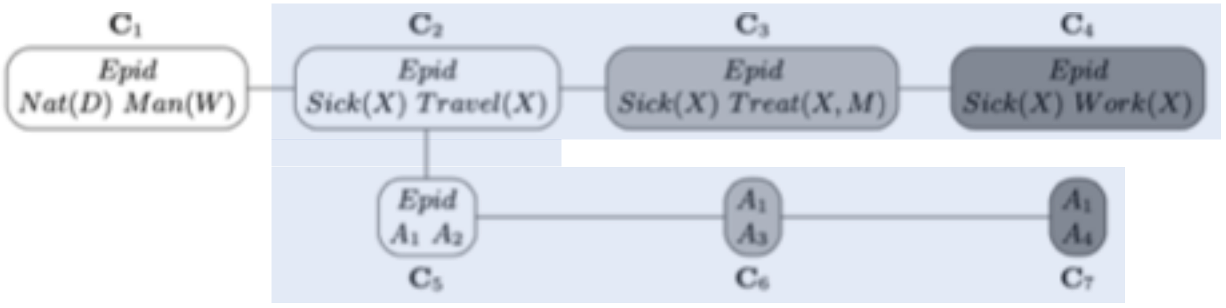
- E.g., add a parfactor $g^+ = \phi\big(A_4, Work(X)\big)$
  - $Work(X)$ contained in $\boldsymbol{C}_4$, A_4 contained in $\boldsymbol{C}_7$,
  - Subgraph of $\boldsymbol{C}_4$ and $\boldsymbol{C}_7$



(a) Before adjusting

Adjustment Strategies

- Find a set of parclusters N such that $\boldsymbol{A}^{old} \subseteq rv(N)$
  - Subgraph J_N spanned by N
1. Merge all parclusters in J_N
   - Creates one very large parcluster
2. Add $\boldsymbol{A}^{old}$ to all parclusters in J_N
   - E.g., add $A_4, Work(X)$ to all parclusters in the subgraph
   - Might be able to save adding some PRVs, depending on which PRVs appear in which parcluster
     - E.g., enough to add one of $A_4, Work(X)$ to the path

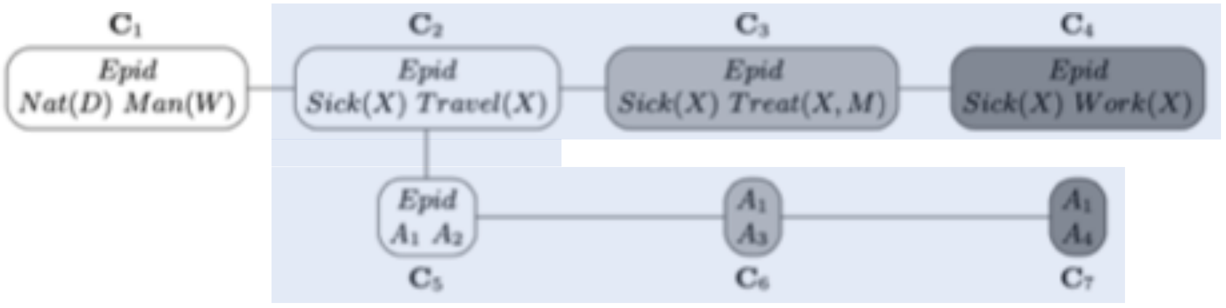


(a) Before adjusting

Adjustment Strategies

- Find a set of parclusters N such that $\boldsymbol{A}^{old} \subseteq rv(N)$
  - Subgraph J_N spanned by N

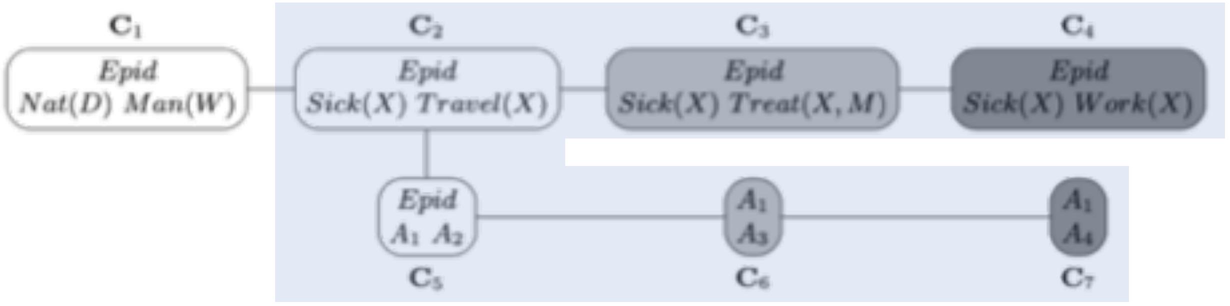
3. While $|N| > 1$
   - Take two parclusters out of N with P the path between them
   - While $length(P) > 1$
     - Take the ends out of P and merge them → creates a cycle
     - Check if you can delete an edge on the cycle while the running intersection property still holds → If so, delete the edge + stop
   - Maximum parcluster size doubles at most



(a) Before adjusting

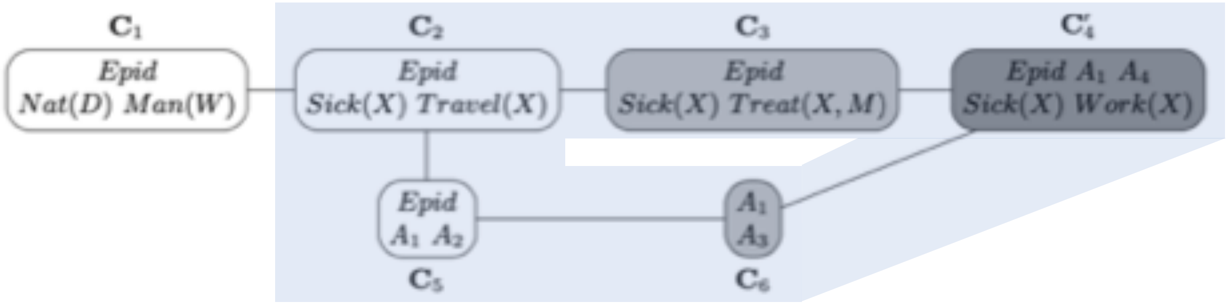
Adjusting FO Jtrees: Example

- To add a parfactor $g^+ = \phi(A_4, Work(X))$
  - $Work(X)$ contained in $\boldsymbol{C}_4$, A_4 contained in $\boldsymbol{C}_7$, path



(a) Before adjusting

- Start merging at ends of path

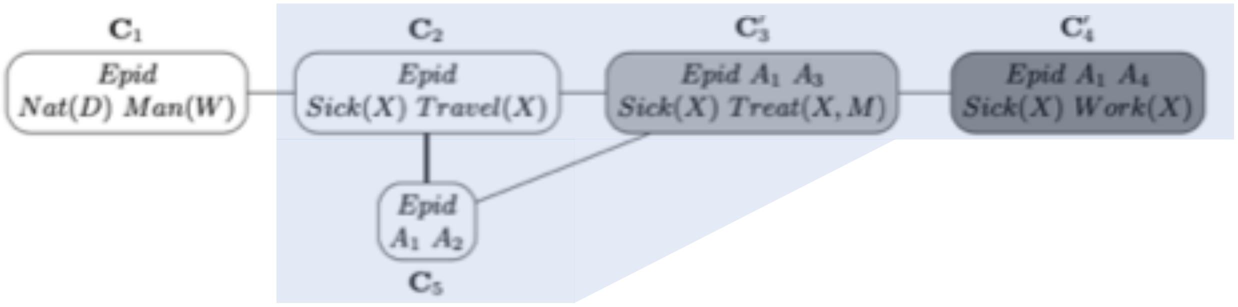


(b) After first merging

  - Cannot break cycle without violating run. intersect. prop.

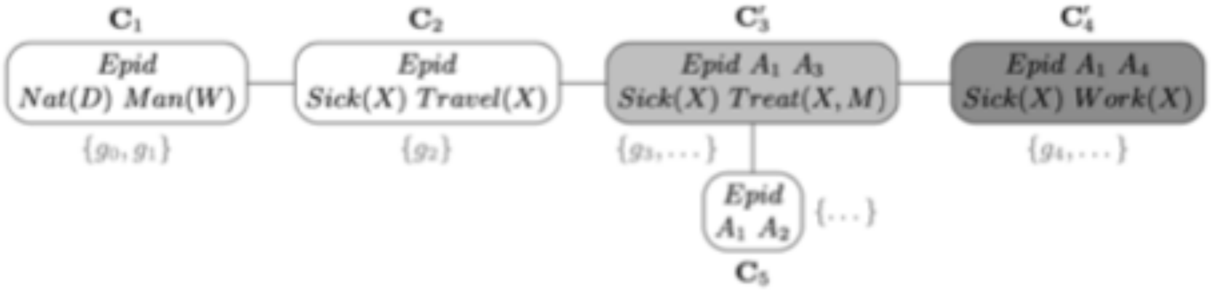
Adjusting FO Jtrees: Example

- Merge next two parclusters on path ends



(c) After second merging

- Can break cycle by deleting edge between $\boldsymbol{C}_2$, $\boldsymbol{C}_5$



(d) After breaking cycle

  - Stop

Adjusting FO Jtrees

- After adjustment: $\exists \boldsymbol{C}_i : \boldsymbol{A}^{old} \subseteq \boldsymbol{C}_i$
- Proceed as before
  1. If $rv(g^+) \subseteq \boldsymbol{C}_i$: add g^+ to G_i
     - $\boldsymbol{A}^{new} = \emptyset$
  2. If $rv(g^+) \supset \boldsymbol{C}_i$: extend $\boldsymbol{C}_i$ with $\boldsymbol{A}^{new}$, add g^+ to G_i
     - $\boldsymbol{A}^{old} = \boldsymbol{C}_i$
  3. If $\boldsymbol{A}^{old} \subset \boldsymbol{C}_i \wedge \boldsymbol{A}^{new} \cap \boldsymbol{C}_i = \emptyset$, build $\boldsymbol{C}_k = rv(g^+)$ as neighbour to $\boldsymbol{C}_i$, add g^+ to G_k

Adding a Parfactor: Procedure

- Input: new parfactor g^+ added to model G with FO jtree $J = (V, E)$
  - $\boldsymbol{A}^{old} = rv(g^+) \cap rv(G)$: known PRVs
  - $\boldsymbol{A}^{new} = rv(g^+) \setminus rv(G)$: new PRVs
- Adjust J such that $\exists \boldsymbol{C}_i : \boldsymbol{A}^{old} \subseteq \boldsymbol{C}_i$
- If $\boldsymbol{A}^{new} = \emptyset$, i.e., $rv(g^+) = \boldsymbol{A}^{old} \subseteq \boldsymbol{C}_i$
  - Add g^+ to G_i, mark G_i as changed
- Else if $\boldsymbol{A}^{old} = \boldsymbol{C}_i$, i.e., $rv(g^+) \supset \boldsymbol{C}_i$
  - Extend $\boldsymbol{C}_i$ with $\boldsymbol{A}^{new}$
  - Add g^+ to G_i, mark G_i as changed
- Else, i.e., $rv(g^+) \not\supset \boldsymbol{C}_i \wedge rv(g^+) \nsubseteq \boldsymbol{C}_i \wedge rv(g^+) \cap \boldsymbol{C}_i \neq \emptyset$
  - Create new parcluster $\boldsymbol{C}_k = rv(g^+)$ with $G_k = \{g^+\}$, mark G_k as changed
  - Add $\boldsymbol{C}_k$ to V and $\{i, k\}$ to E

Addition

- (Afterwards, perform adaptive message passing)

Deleting a Parfactor

- Parfactor $g^{-} \in G$ to delete from model G with FO jtree $J = (V, E)$
- Even after deleting g^{-}, J has to fulfil the FO jtree properties, i.e., after deleting g^{-}
    - Property 1 holds: $\forall \boldsymbol{C} \in V : \boldsymbol{C} \subseteq rv(G)$
        - Deleting a parfactor might delete a PRV from G
    - FO jtree should still be minimal
- Goals for deletion procedure:
    1. Maintain FO jtree properties
    2. Keep FO jtree minimal

1. $\forall \boldsymbol{C} \in V : \boldsymbol{C} \subseteq rv(G)$
2. $\forall g \in G : \exists \boldsymbol{C} \in V : rv(g) \subseteq \boldsymbol{C}$
3. If $\exists A \in rv(G) : A \in \boldsymbol{C}_i \land A \in \boldsymbol{C}_j$ with $\boldsymbol{C}_i, \boldsymbol{C}_j \in V$, then $\forall \boldsymbol{C}_k \in V$ on the path between $\boldsymbol{C}_i, \boldsymbol{C}_j : A \in \boldsymbol{C}_k$

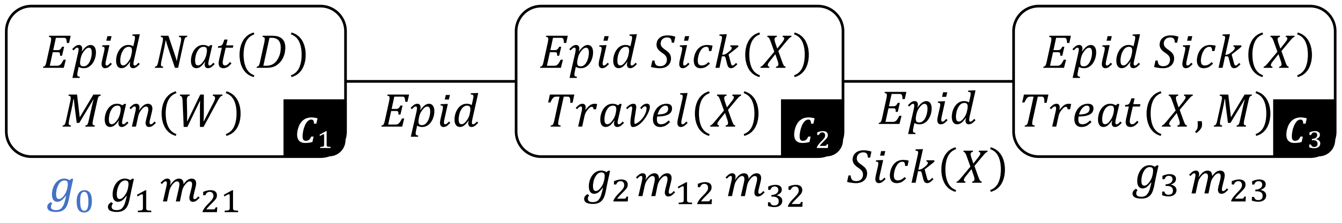
Deleting a Parfactor: Procedure

- Parfactor $g^- \in G$ to delete from model G with FO jtree $J = (V, E)$
- Find parcluster $\boldsymbol{C}_i$ such that $g^- \in G_i$
- Remove g^- from G_i, mark G_i
  - Makes PRVs $rv(g^-) \setminus rv(G_i)$ superfluous in terms of the local model
- Check for each $A \in \left(rv(g^-) \setminus rv(G_i)\right)$ if A needed for running intersection property
  - If not, delete A
- If any PRV deleted, check if $\boldsymbol{C}_i$ now a subset of a neighbour $\boldsymbol{C}_j$
  - If so, merge $\boldsymbol{C}_i, \boldsymbol{C}_j$ and mark resulting local model

Deletion

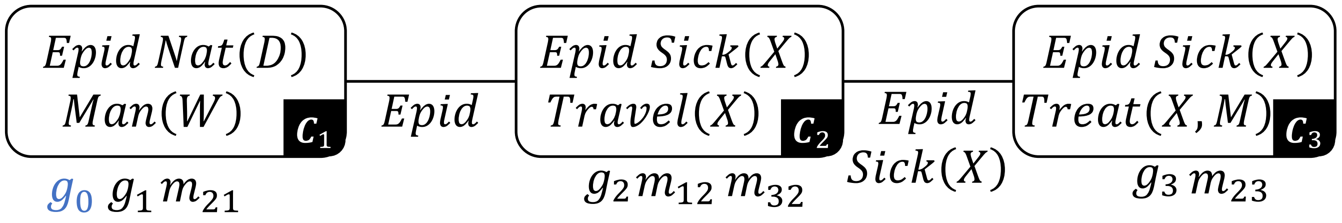
Deleting a Parfactor: Example

- Assume another parfactor $g_0 = \phi(Epid)$ at $\boldsymbol{C}_1$
- Consider removing g_0
  - Does not make any PRVs superfluous w.r.t. G_1 as g_1 contains all PRVs of the parcluster
  - Simply remove g_0 from G_1
- Consider removing g_1
  - Makes PRVs $Nat(D), Man(W)$ superfluous w.r.t. G_1 as g_0 only contains $Epid$
  - $Nat(D), Man(W)$ also not necessary to maintain the running intersection property
  - Remove g_0 from G_1, delete $Nat(D), Man(W)$
  - Now, $\boldsymbol{C}_1$ is a subset of $\boldsymbol{C}_2$ → merge $\boldsymbol{C}_1$, $\boldsymbol{C}_2$



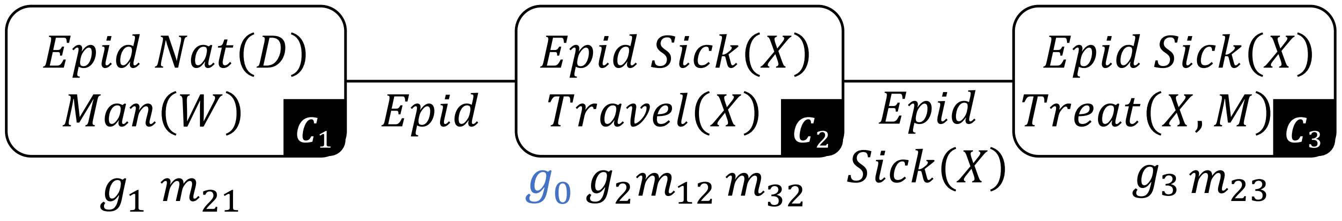
Deleting a Parfactor: Example

- Assume another parfactor $g_0 = \phi(Epid, Nat(D))$ at $\boldsymbol{C}_1$
- Consider removing g_1
  - Makes PRV $Man(W)$ superfluous w.r.t. G_1 as g_0 only contains $Epid, Nat(D)$
  - $Man(W)$ also not necessary to maintain the running intersection property
  - Remove g_1 from G_1, delete $Man(W)$
  - $\boldsymbol{C}_1$ is not a subset of $\boldsymbol{C}_2$ → no merging



Deleting a Parfactor: Example

- Assume another parfactor $g_0 = \phi\big(Sick(X), Travel(X)\big)$ at $\boldsymbol{C}_2$
- Consider removing g_2
  - Makes PRV $Epid$ superfluous w.r.t. G_2 as g_0 only contains $Sick(X), Travel(X)$
  - But: $Epid$ necessary to maintain the running intersection property
  - Only remove g_2 from G_2



UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

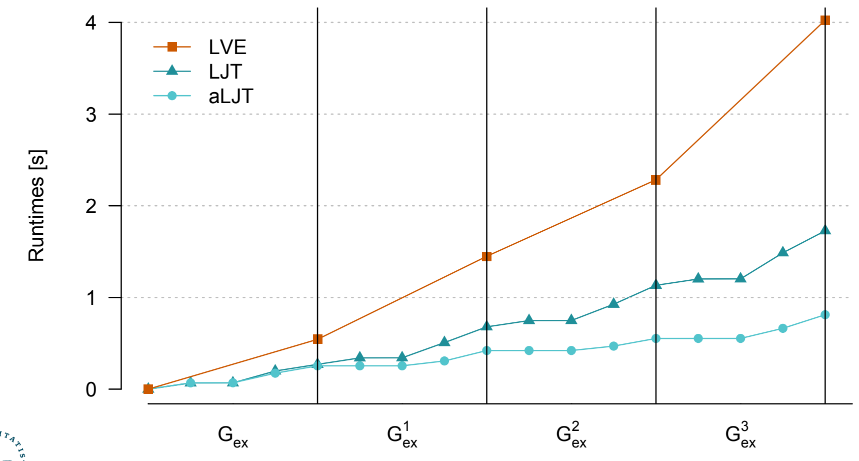
LJT for Adaptive Inference: Algorithm

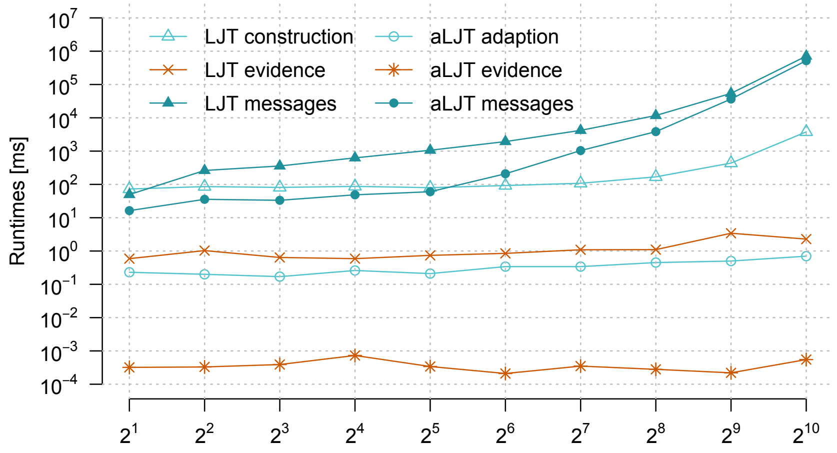
```
**aLJT**(FO jtree $J$ of model $G$ with evidence $\boldsymbol{E}$, queries $\{Q_i\}_{i=1}^n$, evidence changes $\Delta_E$, model changes $\Delta_G$)
    Adapt $J$ to $\Delta_G$ (with addition, deletion)
    for each parcluster $\boldsymbol{C}_i$ in $J$ do
        if $\boldsymbol{C}_i$ $\boxed{\text{marked}}$ or affected by $\Delta_E$ then
            Handle or adapt evidence at $\boldsymbol{C}_i$, mark $\boldsymbol{C}_i$
    while $\exists \boldsymbol{C}_i$ ready to compute message $m_{ij}$ to $\boldsymbol{C}_j$ do
        if $\boldsymbol{C}_i$ marked, has a marked message, or $\boldsymbol{C}_j$ new then
            Send newly computed $m_{ij}$
        else
            Send empty message $m_{ij} = \emptyset$
        Mark $m_{ij}$ at $\boldsymbol{C}_j$ accordingly (valid/changed)
    Answer queries with query terms $\{Q_i\}_{i=1}^n$ in $J$
```

If local model changed, then evidence might need to be entered again

Runtimes

- Get to the point of answering queries again as fast as possible
- Left: Accumulated runtimes over three changes
  - Vertical lines mark a change
  - Dots mark steps of LJT versions
- Right: runtimes for each step of LJT/aLJT after a model change (model size doubles with each change)



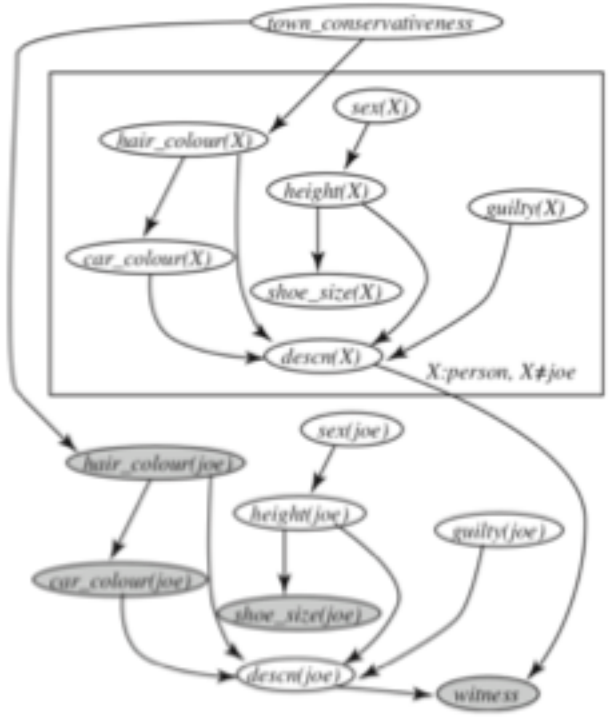


Model Changes

- Adaptive inference reasonable if effect of changes does not change FO jtree structure
  - Always possible to save at least one message calculation up to half of the message pass
- Adaptive inference only reasonable if effect of changes FO jtree structure only locally
  - I.e., limited number of parclusters affected
  - If changes to model are too many or too drastic, FO jtree may degenerate
    - Restart from scratch and build a new FO jtree
- Similar considerations for circuits and FOKC
  - Only repeat weight propagation in branches with changes

Leaving a specific domain

What happens if **domains change**?



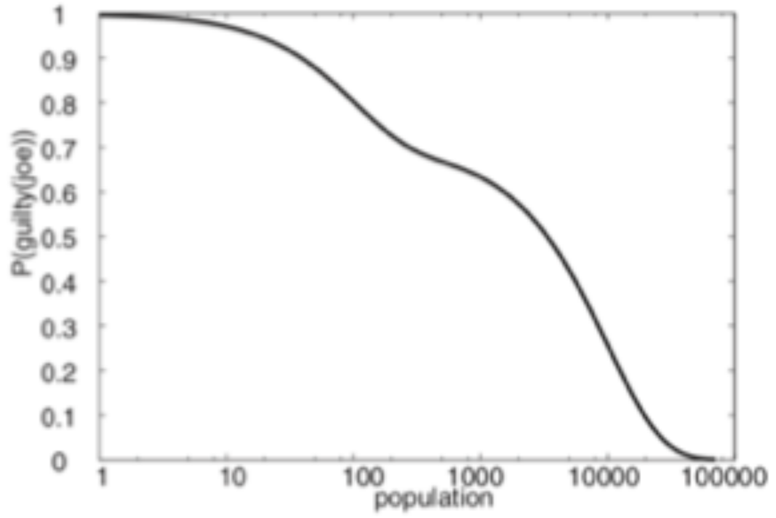
Known Domains

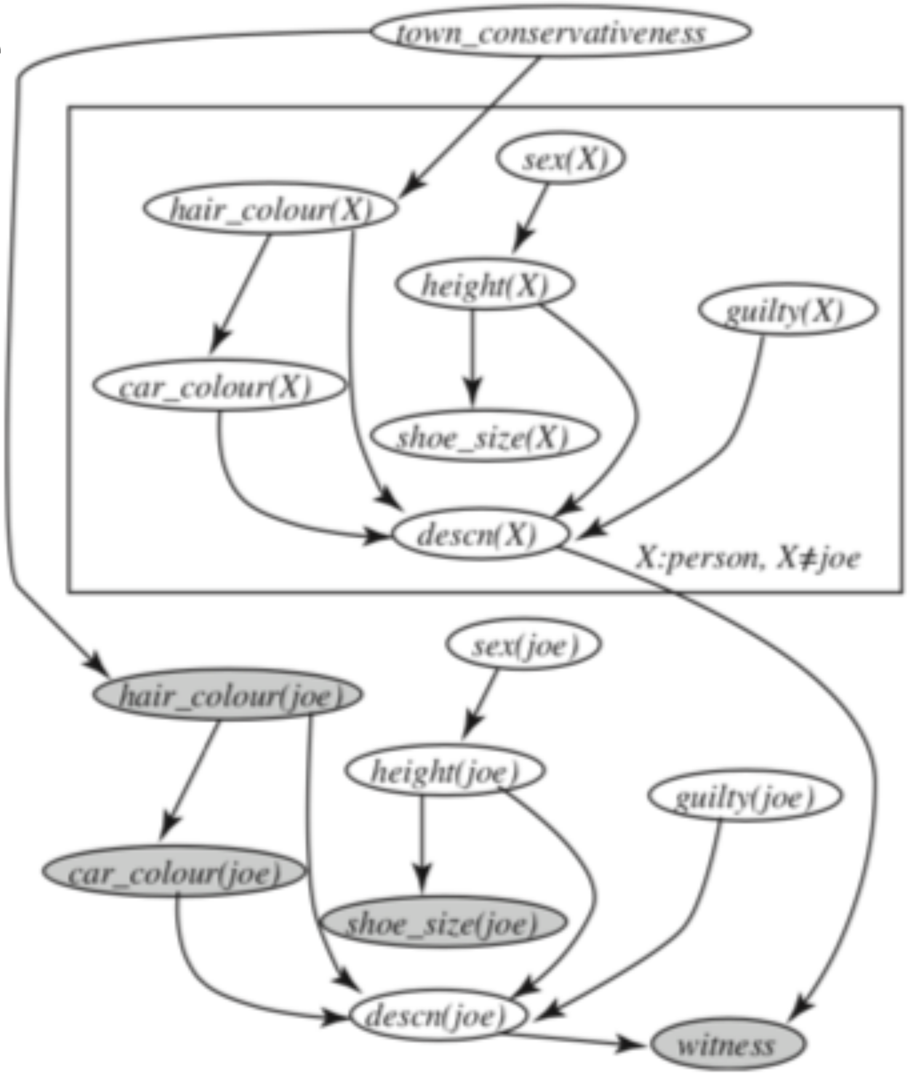
- Grounding semantics is only defined given specific domains for logical variables
  - Evidence for known constants
  - Queries reference known constants
- Also, models usually learned on a specific domain

- What if…
  - domains change?
  - domains are unknown?

Changing Domains

- Keep semantics as before
  - Assume that parfactors accurately describe world
- Posterior probabilities change depending on domain sizes
  - Example by Poole (2003)

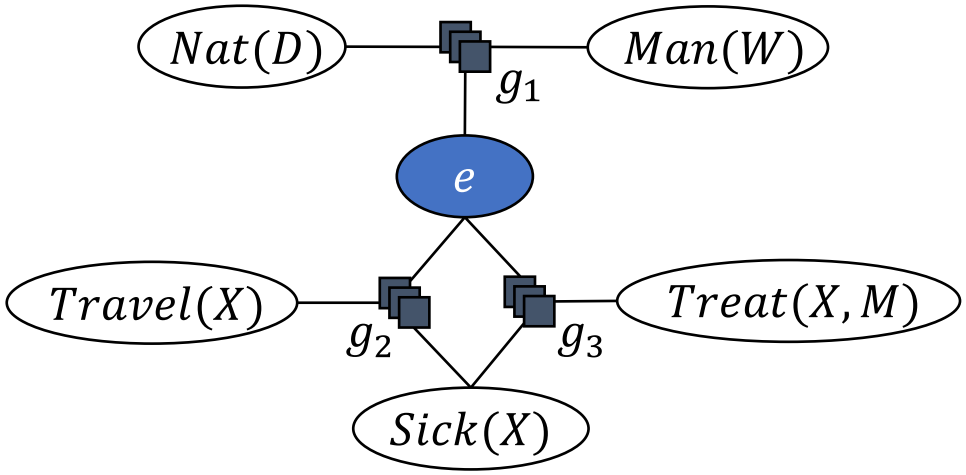




David Poole. First-order Probabilistic Inference. In *IJCAI-03 Proc. of the 18th Internat. Joint Conf. on AI*, 2003.

… Without Effects

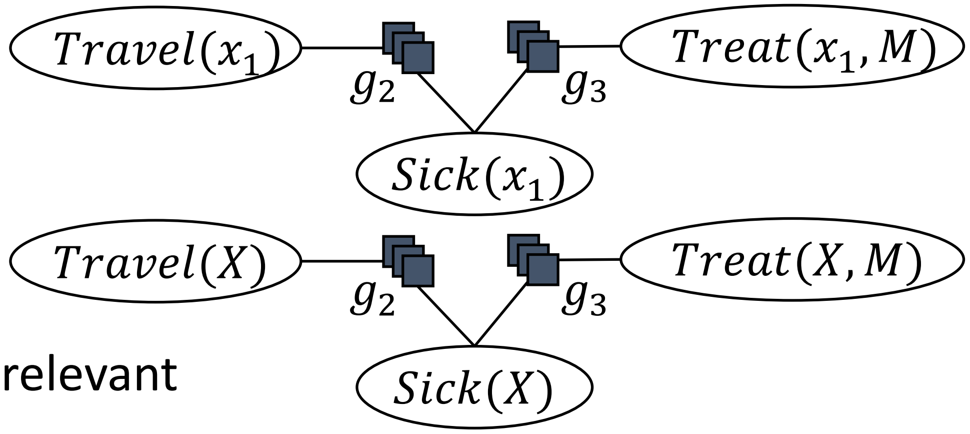
- (Conditional) Independence
  PRVs, containing logical variables $\boldsymbol{X}$, that are (conditionally) independent from query terms → domains of $\boldsymbol{X}$ have no influence on query results
- E.g., given $Epid = e$,
  - $\mathcal{D}(D)$ and $\mathcal{D}(W)$ do not matter for queries regarding $Travel$, $Sick$, and $Treat$
  - $\mathcal{D}(X)$ and $\mathcal{D}(M)$ do not matter for queries regarding Nat and Man



→ Partly invariant under increasing domain sizes

… Without Effects

- A simple case of so-called projectivity
  - After shattering, query terms are independent of model parts containing logical variables $\boldsymbol{X}$
    $\rightarrow$ domains of $\boldsymbol{X}$ have no influence on query results
    - Depends on model structure
    - More by Jaeger and Schulte (2018)
- E.g., $P(Sick(x_1))$
  - $\mathcal{D}(X) = \{x_1, \dots, x_n\}$
  - After shattering:
    - $\mathcal{D}(X) = \{x_2, \dots, x_n\}$
    - Upper part independent from lower part; $\mathcal{D}(X)$ irrelevant

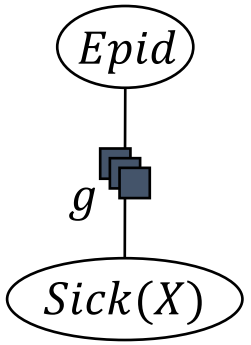


$\rightarrow$ Partly invariant under increasing domain sizes

Manfred Jaeger and Oliver Schulte. Inference, Learning, and Population Size: Projectivity for SRL Models. In *StaRAI-18 Workshop on Statistical Relational Artificial Intelligence*, 2018.

Growing Domain Sizes

- Let domain size n grow
  - With grounding semantics, posteriors change
    - Can lead to extreme behaviour in the posteriors
- Example: $Epid$ gets more and more neighbours with n rising



$$P(Epid) \propto \left(\sum_{s \in \mathcal{R}(Sick(X))} g(Epid, Sick(x) = s) \right)^n$$

$$= \left(g'(Epid)\right)^n = g''(Epid) = g^\alpha(Epid)$$

$Epid$	g'
$false$	a
$true$	b

$Epid$	g''
$false$	a^n
$true$	b^n

$Epid$	g^α
$false$	$\frac{a^n}{a^n + b^n}$
$true$	$\frac{b^n}{a^n + b^n}$

$$\rightarrow \overbrace{\frac{1}{1 + \left(\frac{b}{a}\right)^n}}^{\text{Sigmoid function}} \xrightarrow{n \to \infty} \begin{cases} 0 & b > a \\ 1 & a > b \end{cases}$$

David Poole, David Buchman, Seyed Mehran Kazemi, Kristian Kersting, and Sriraam Natarajan. Population Size Extrapolation in Relational Probabilistic Modeling. In *SUM-14 Proceedings of the 8th International Conference on Scalable Uncertainty Management*, 2014.

Growing Domain Sizes

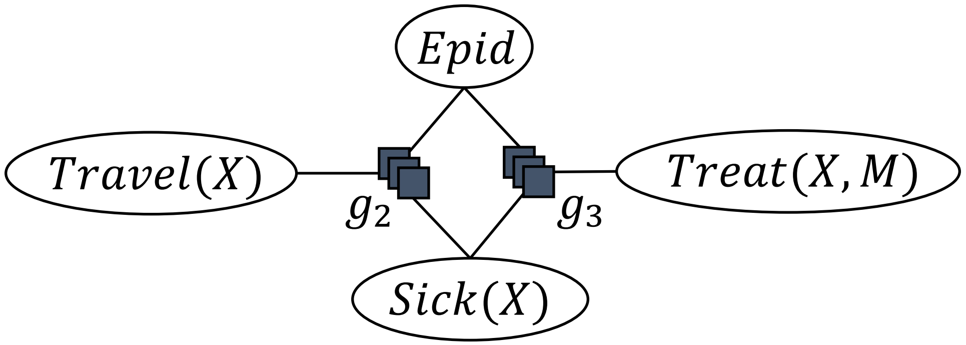
- How to avoid extreme behaviour?

→ Adapt values in model dependent on domain size

- Approach for MLNs: Domain-size aware MLNs
  - Assume predicates $P_1, \dots, P_m$ occur in a FOL formula F
    - Count number of connections c_j for each predicate P_j given *new* domains
    - Build a connection vector $[c_1, \dots, c_m]$
    - Choose $\max_{c_i}[c_1, \dots, c_m]$ as scaling-down factor
      - Instead of $\max$, other functions possible
      - Works best if the values in $[c_1, \dots, c_m]$ do not vary that much
  - Given an MLN with a set of formulas F_i with weights w_i
    - Rescale each w_i with scaling-down factor s_i computed for F_i as $\frac{w_i}{s_i}$
- Analogous approach possible for parfactors

Unknown Domains

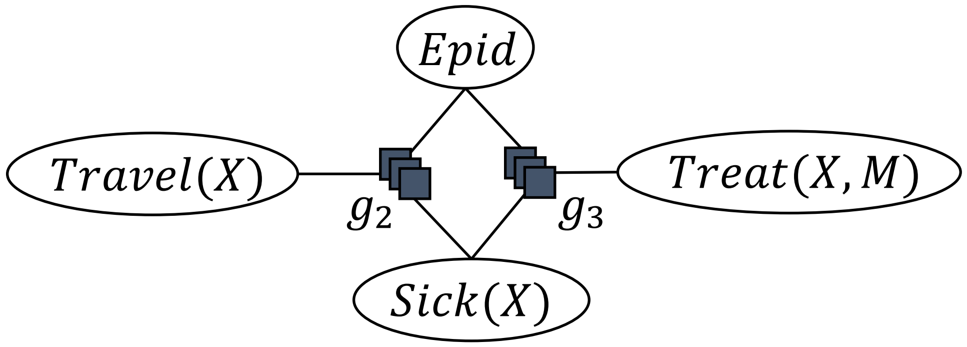
- General domains of logical variables, e.g.,
  - $\mathcal{D}(X) =$ ~~$\{alice, eve, bob\}$ or~~
  - $\mathcal{D}(X) =$ ~~$\{x_1, \dots, x_n\}$~~
- Constraint C_i in each parfactor g_i, e.g.,
  - $C_3 = \big((X, M),$ ~~$\mathcal{D}(X) \times \mathcal{D}(M)$~~$\big)$
- ~~Based on constraints, grounding semantics apply~~
  - ~~Lifted algorithms work~~



Template Model

- Assume that local potential functions accurately describe behaviour
- Template model:
  Parfactors with empty constraints
  - $\tilde{G} = \{\tilde{g}_i\}_{i=1}^n$
    - $\tilde{g}_i = \phi_i(\mathcal{A}_i)_{|\tilde{C}_i}$
      - $\tilde{C}_i = (\mathcal{X}, \perp)$
      - $\mathcal{X}$ a sequence of the logvars in $\mathcal{A}_i$
  - E.g., in $G = \{g_i\}_{i=2}^3$
    - $\tilde{C}_2 = ((X), \perp)$
    - $\tilde{C}_3 = ((X, M), \perp)$

Use $\perp$ instead of $\emptyset$ as $\emptyset$ could be a valid constraint set in some scenarios



Constraint Program

- Constraint program $\mathcal{C}$:
  Generate constraints, possibly accompanied by a probability, for a template model $\tilde{G} = \{\tilde{g}_i\}_{i=1}^n$ given a domain:

$$\boldsymbol{C} = \left\{ \left(\left\{ C_{j,i} \right\}_{i=1}^{n}, p_j \right) \right\}_{j=1}^{m}$$

  - $C_{\cdot,i}$ a constraint to replace $\tilde{C}_i$ in $\tilde{G}$
    - Replacing the empty constraints in $\tilde{G}$ with the constraints in a constraint world leads to a standard model G
  - Each $\left(\left\{ C_{j,i} \right\}_{i=1}^{n}, p_j \right)$ called a *constraint world*
    - Each constraint world can replace the constraints in $\tilde{G}$
  - Leads to a set of models $\boldsymbol{G} = \left\{ \left(G_j, p_j \right) \right\}_{j=1}^{m}$ where
    - G_j is $\tilde{G}$ but the empty constraints are replaced by $\left\{ C_{j,i} \right\}_{i=1}^{n}$

Constraint Program

- Constraint program $\mathcal{C}$:
  Generate constraints, possibly accompanied by a probability, for a template model $\tilde{G} = \{\tilde{g}_i\}_{i=1}^n$ given a domain:
  $$\boldsymbol{C} = \left\{\left(\{C_{j,i}\}_{i=1}^n, p_j\right)\right\}_{j=1}^m$$
  - If no probabilities given per constraint world
    - Either just a set of sets (set of constraint worlds without p_j)
    - Or provide an equal distribution over all worlds: $p_j = \frac{1}{m}$
  - If program yields one world, probability of the one constraint world is 1:
    $$\boldsymbol{C} = \left\{\left(\{C_{j,i}\}_{i=1}^n, 1\right)\right\}_{j=1}^{m=1} \rightarrow \{C_{1,i}\}_{i=1}^n = \{C_i\}_{i=1}^n$$
    - Yields one model G with constraints $\{C_i\}_{i=1}^n$

Tanya B and Ralf Möller. Exploring Unknown Universes in Probabilistic Relational Models. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, 2019.

Constraint Program: Example

- E.g., using *probabilistic* Datalog (Fuhr, 1995):
  - Set of rules (Horn clauses) and facts *that are possibly weighted by probability*
  - Free variable occurring in $\mathcal{C}^{DL}$: X
    - Provide a domain for X, e.g., $\mathcal{D}(X) = \{alice, eve, bob\}$ by adding facts (*no weight = always true*)

```
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

$\mathcal{C}^{DL}$

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
```

Mutually exclusive (only one is true at a time)

Norbert Fuhr: Probabilistic Datalog. In: *SIGIR-95 Proceedings of the 18th Annual International ACM SIGIR conference on Research and Development on Information Retrieval*, 1995.

Constraint Worlds: Example

- Generate tuples for constraints by asking query
  - Queries can contain free variables
    - Answers provide valid groundings of free variables associated with a probability
  - E.g.,
    - $C_2 = \big((X), \perp\big)$: `?- instance_of_X(X)`
    - $C_3 = \big((X, M), \perp\big)$: `?- element_of_C3(X,M)`

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

Norbert Fuhr: Probabilistic Datalog. In: *SIGIR-95 Proceedings of the 18th Annual International ACM SIGIR conference on Research and Development on Information Retrieval*, 1995.

Constraint Worlds: Example

- $C_2 = ((X), \perp)$: `?- instance_of_X(X)`
  - Query returns the following facts:
    - Return those groundings of the query that evaluate to true

```
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

Constraint Worlds: Example

- $C_3 = \big((X,M),\bot\big)$: `?- element_of_C3(X,M)`
  - Query returns the following facts using `0.7 pair(t1,t2)`:

```
0.7 element_of_C3(alice,t1).
0.7 element_of_C3(alice,t2).
0.7 element_of_C3(eve,t1).
0.7 element_of_C3(eve,t2).
0.7 element_of_C3(bob,t1).
0.7 element_of_C3(bob,t2).
```

Leads to three sets of facts of which only one set is true with a probability as given by the fact

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

Mutually exclusive (only one is true at a time)

Constraint Worlds: Example

- Three constraint worlds $\boldsymbol{C} = \left\{ \left(\left\{ C_{j,i} \right\}_{i=1}^{n}, p_j \right) \right\}_{j=1}^{m=3}$
  - $j = 1$ using `0.7 pair(t1,t2).`: $\left(\left\{ C_{1,i} \right\}_{i=2}^{3}, p_1 \right)$
    - $p_1 = 0.7$
    - $C_{1,2} = \left((X), \{a, e, b\} \right)$
    - $C_{1,3} = \left((X, M), \{(a, t1), (a, t2), (e, t1), (e, t2), (b, t1), (b, t2)\} \right)$
    - with $a = alice, e = eve, b = bob$

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

Constraint Worlds: Example

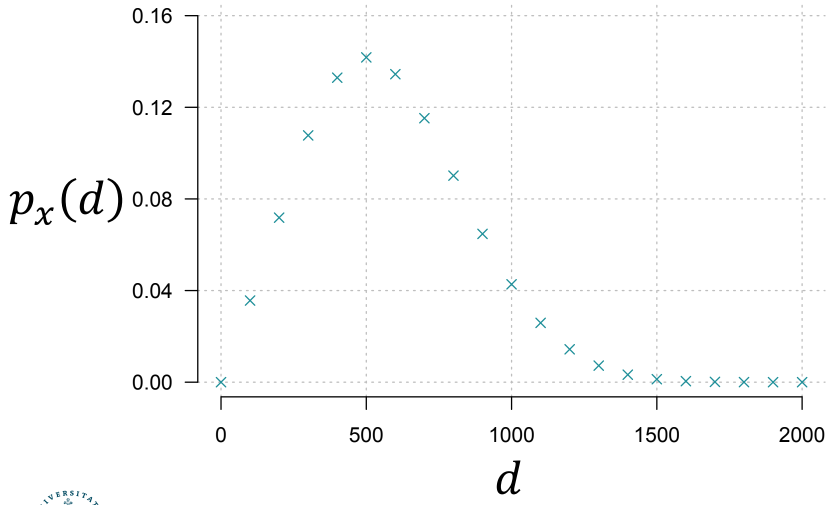
- $j = 2$ using `0.2 pair(t2,t3).`: $\left(\{C_{2,i}\}_{i=2}^{3}, p_2\right)$:
    - $p_2 = 0.2$
    - $C_{2,2} = \left((X), \{a, e, b\}\right)$
    - $C_{2,3} = \left((X, M), \{(a, t2), (a, t3), (e, t2), (e, t3), (b, t2), (b, t3)\}\right)$
- $j = 3$ using `0.1 pair(t1,t3).`: $\left(\{C_{3,i}\}_{i=2}^{3}, p_3\right)$:
    - $p_3 = 0.1$
    - $C_{3,2} = \left((X), \{a, e, b\}\right)$
    - $C_{3,3} = \left((X, M), \{(a, t1), (a, t3), (e, t1), (e, t3), (b, t1), (b, t3)\}\right)$

$a = alice$
$e = eve$
$b = bob$

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3).
instance_of_X(alice).
instance_of_X(eve).
instance_of_X(bob).
```

Domain Worlds

- Specify or generate possible domains
- Encode assumptions like
  - Small domains more likely than large domains
  - Only rough counts necessary (500 vs. 499 vs. 501)
- For X, e.g.,
  - Beta-binomial distribution ($\alpha = 6, \beta = 15$) referred to as $p_x(d)$ with d as the domain size of X



$$\boldsymbol{D} = \left\{\left(\{x_i\}_{i=1}^{d}, p_x(d)\right)\right\}_{d=100, d+=100}^{2000}$$

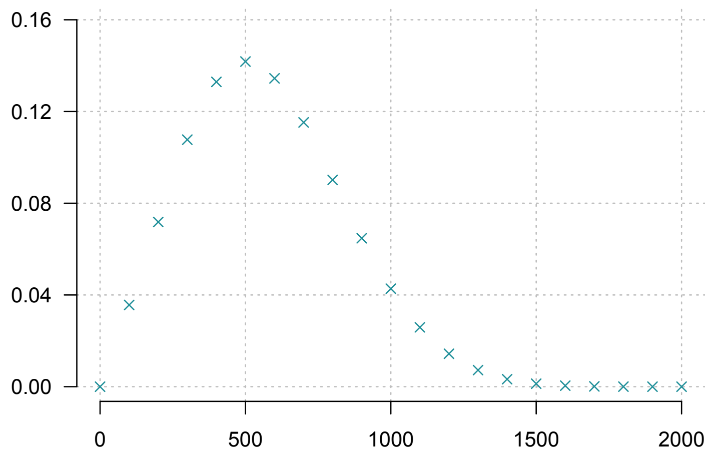
- Each as input to constraint program $\mathcal{C}^{DL}$

Yields 20 domains with probability > 0

Tanya B and Ralf Möller. Exploring Unknown Universes in Probabilistic Relational Models. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, 2019.

Interworkings

- Distribution over domains



Together, they yield $20 \cdot 3$ constraint worlds, each with probability > 0

If *both* domain and constraint worlds are associated with probabilities, *multiply* both probabilities, assuming the probabilities are independent.

- … as input to probabilistic constraint program

$\mathcal{C}^{DL}$

```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3)
```

Tanya B and Ralf Möller. Exploring Unknown Universes in Probabilistic Relational Models. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, 2019.

Filtering

- Some probabilities can be or become very low
- Filtering based on probabilities; e.g.,
  - Threshold t or
  - Keep only those models whose probabilities make up, e.g., 95% of the distribution around its mean or maximum value
- *Cascading* filtering
  1. Filter domain worlds
  2. Filter constraint worlds resulting from (remaining) domain worlds

Groundings-based Semantics

Inputs

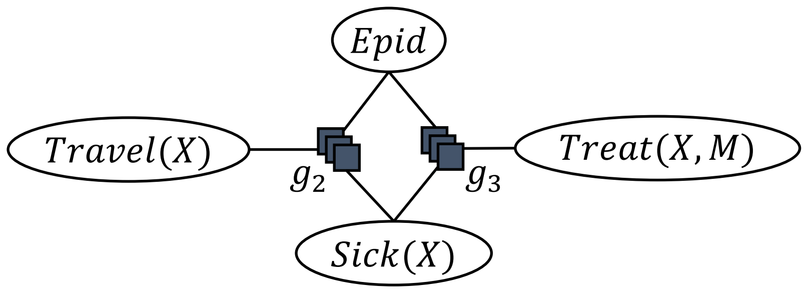
- Template model
    - Empty constraints
- Constraint program
    - Fill empty constraints given a domain world
    - Can generate a probability distribution over models
- Domain worlds
    - Generate possible worlds as input to constraint program
    - Can be a probability distribution over domains
- Optionally, threshold t

Approach

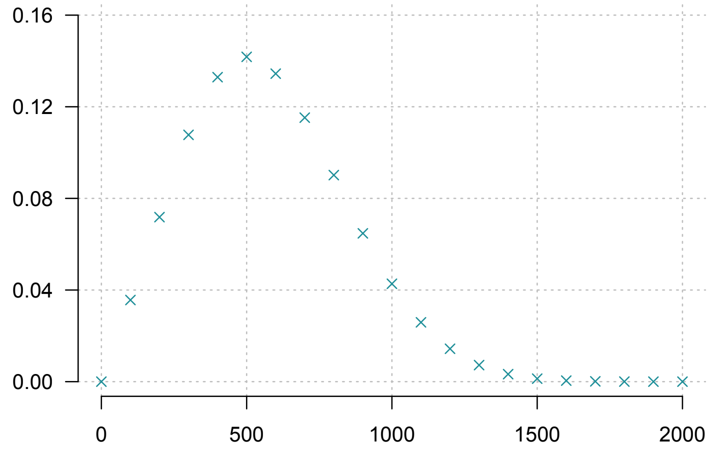
- Generate a set of possible models
    - Can be a probability distribution over possible models
    - Within model: grounding semantics apply
        - Lifted algorithms work again
- Reasoning over possible models
    - New query types

Example: Inputs

- Template model



- Domain worlds



- Constraint program

$\mathcal{C}^{DL}$

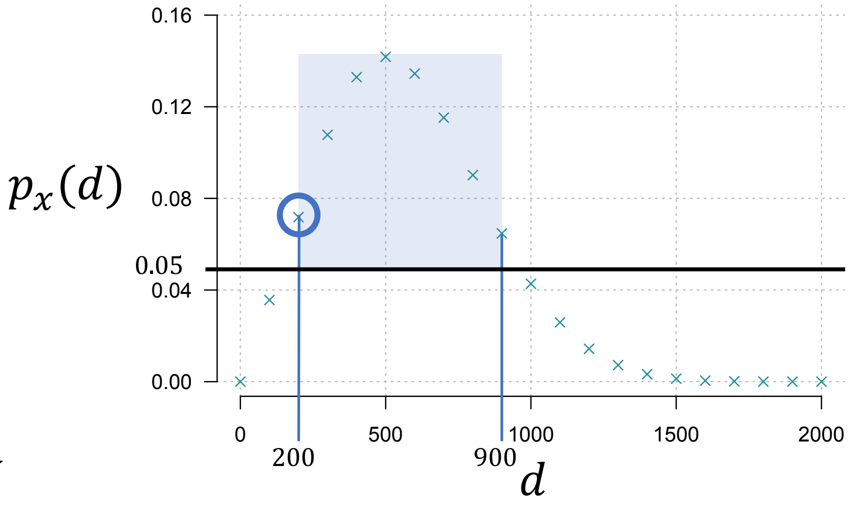
```
element_of_C3(X,Y1) :- linked(X,Y1,Y2).
element_of_C3(X,Y2) :- linked(X,Y1,Y2).
linked(X,Y1,Y2) :- instance_of_X(X) & pair(Y1,Y2).
0.7 pair(t1,t2).
0.2 pair(t2,t3).
0.1 pair(t1,t3)
```

- Filtering with $t = 0.05$

Example

- Filtering with $t = 0.05$ brings domain worlds down to 8 domain worlds

$$\left\{\left(\{x_i\}_{i=1}^{d}, p_x(d)\right)\right\}_{d=200, d+=100}^{900}$$

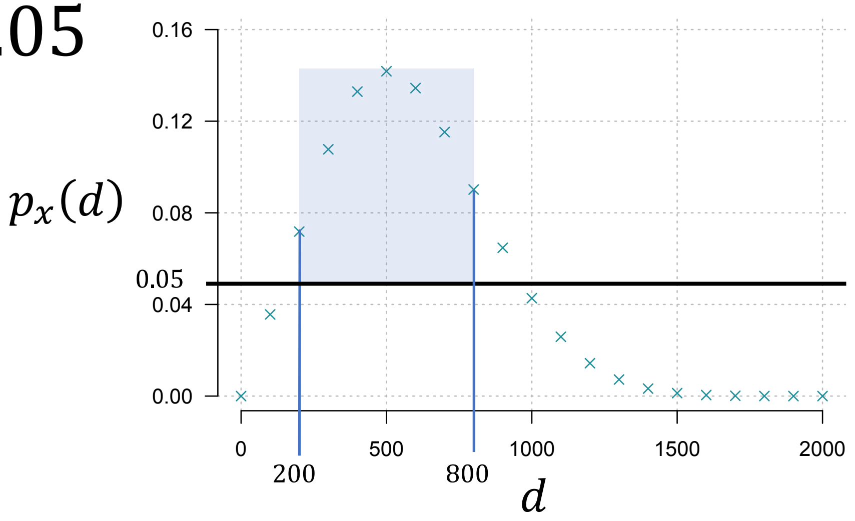


  - E.g., for $d = 200$
    - $\left(\{x_i\}_{i=1}^{200}, p_x(200)\right)$
    - $p_x(200) = 0.07182$
    - x_i the constants for X

- Generate constraint worlds (3 per domain, i.e., 24)
  - E.g., for $d = 200$ and `0.7 pair(t1,t2)` for $j = 1$:
    - $C_{1,2} = \left((X), \{x_i\}_{i=1}^{200}\right)$
    - $C_{1,3} = \left((X, M), \{x_i\}_{i=1}^{200} \times \{t1, t2\}\right)$
    - Probability: $p_x(200) \cdot 0.7 = 0.050274$

Example

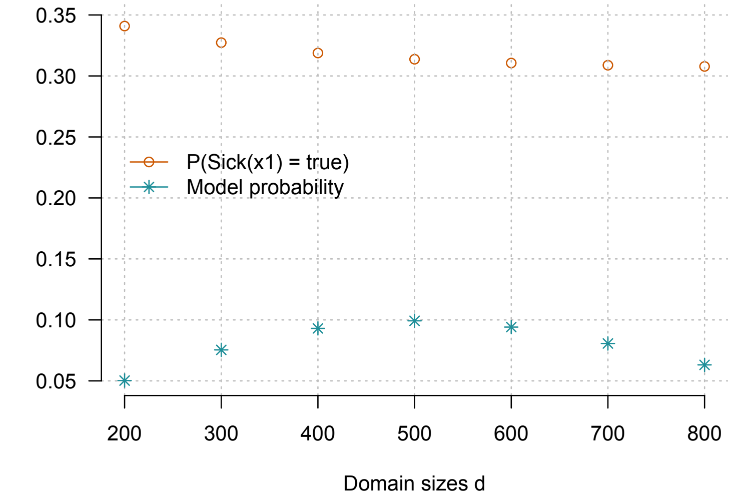
- Filtering again with $t = 0.05$ brings constraint worlds down to 7
  - *Cascaded filtering*
  - Domain worlds that yield constraint worlds with probability $p_x(d) \cdot p_j > 0.05$ for each resulting model:
    - $\left\{\left(\{x_i\}_{i=1}^{d}, p_x(d)\right)\right\}_{d=200, d+=100}^{800}$ and
    - $j = 1$: `0.7 pair(t1,t2)` in $\mathcal{C}^{DL}$
      - $C_{d,1,2} = \left((X), \{x_i\}_{i=1}^{d}\right)$
      - $C_{d,1,3} = \left((X, M), \{x_i\}_{i=1}^{d} \times \{t1, t2\}\right)$



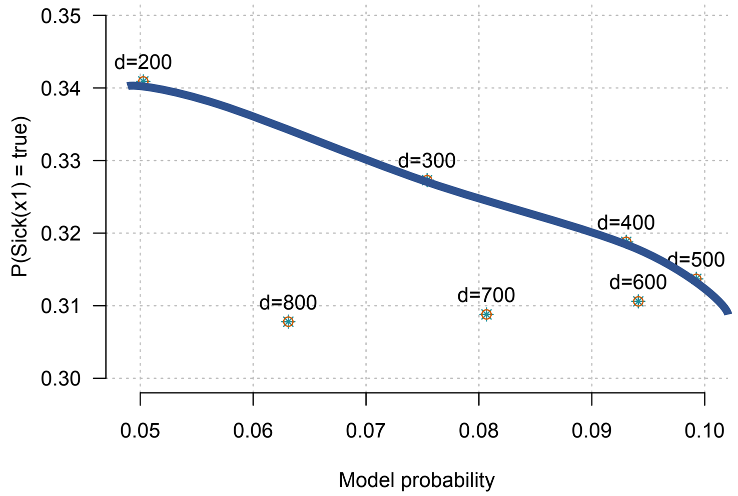
Actually, one does not want to generate all 24 constraint worlds and filter afterwards but stop after the constraint world with probability of domain world $d = 500$ (highest probability among d) and `0.2 pair(t2,t3)` is below t:
$p_x(500) \cdot 0.2 = 0.0284 < 0.05$

New Queries Emerging

- Exploration
  - Model and query probabilities w.r.t.
    - Domain sizes (as in changing domains + grounding semantics)
    - Skyline query



- Model checking
  - E.g., does the probability of
    - an individual being sick decrease with larger domains?
    - an epidemic rise if more people travel?



Tanya B and Ralf Möller. Exploring Unknown Universes in Probabilistic Relational Models. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, 2019.

Most Probable Assignments

Most likely state assignments to (a subset of) the remaining PRVs given evidence

Most Probable Explanation (MPE)

- Given some evidence, what is the most likely state of the remaining randvars?
  - *Assignment query* asking for the most probable assignment to
    - all randvars without evidence
      - Most probable explanation (MPE)
  - Formally, given a model G representing the full joint P_G and evidence $\{E_j = e_j\}_{j=1}^{m}$, $\boldsymbol{e}$ for short,

$$MPE_G(\boldsymbol{e}) = \underset{\boldsymbol{v} \in \mathcal{R}(\boldsymbol{V}_{|C})}{\operatorname{argmax}} P\left((\boldsymbol{V} = \boldsymbol{v})_{|C} | \boldsymbol{e}\right)$$

    - $\boldsymbol{V}_{|C} = rv(G) \setminus \{E_j\}_{j=1}^{m}$ the PRVs in G without evidence
    - Compared to "probability" query, replaced $\sum$ with argmax as the elimination operation

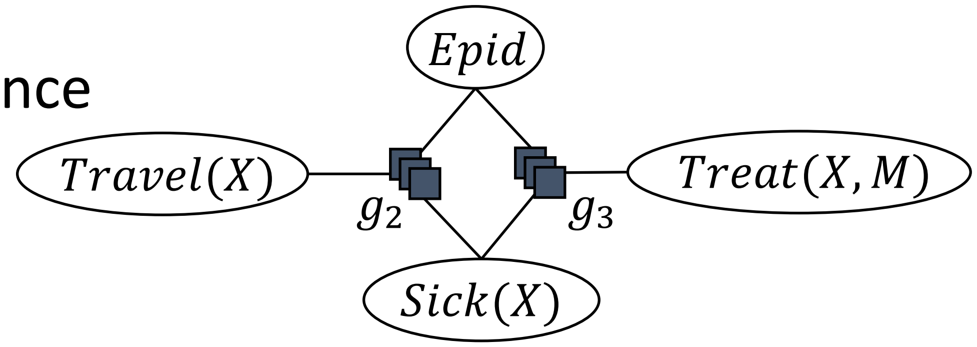
Alexander Philip Dawid. Applications of a General Propagation Algorithm for Probabilistic Expert Systems. *Statistics and Computing*, 2(1):25–36, 1992.
Rina Dechter. Bucket Elimination: A Unifying Framework for Probabilistic Inference. In *Learning and Inference in Graphical Models*, pages 75–104. MIT Press, 1999.

MPE: Semantics

- Based on semantics, could build full joint of G, apply evidence, and then choose the world that has the highest probability assigned
  - E.g., given $Epid = true$, what is the most likely state of $Travel(X), Sick(X), Treat(X, M)$?

$$MPE(epid) = \underset{t,s,tt}{\operatorname{argmax}} P(Travel(X) = t, Sick(X) = s, Treat(X, M) = tt | \boldsymbol{e})$$

- But,
  - ignores factorisation
  - without further evidence on logvars, all instances of PRVs are indistinguishable



(L)VE for MPE Queries

- Replace sum-out operation with a max-out operation
  - Rest stays the same (including needing a heuristics for an elimination order)
  - In knowledge compilation, replace $+$ with $\max$
  - In LVE operator suite, operator $\max-\text{out}$ instead of $\text{sum}-\text{out}$
    - Same input
    - Same preconditions
    - Same postcondition
    - Basically same specification of output
- But: two tasks in one
  - Perform maximisation ($\arg\max$)
  - Store assignments as well ($\arg\max$)

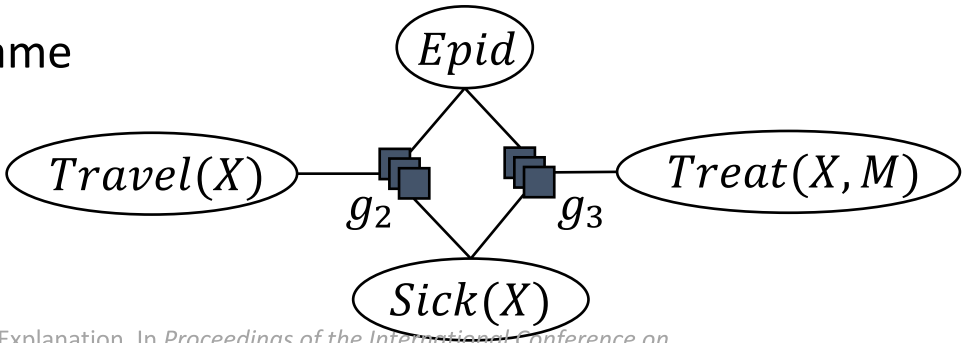
  → Parfactor definition changes slightly to map to potentials and assignments

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. MPE and Partial Inversion in Lifted Probabilistic Variable Elimination. *AAAI-06 Proceedings of the 21st Conference on Artificial Intelligence*, 2006.
Udi Apsel and Ronen I. Brafman. Exploiting Uniform Assignments in First-Order MPE. *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, 2012.
Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

Parfactors for Assignments

- Map each argument setting to its potential and the set of assignments already maxed out
  - Maxing out a P(C)RV means maxing out a set of propositional randvars represented by a P(C)RV
    - All get the same value assigned as they behave identically, therefore have the same assignment leading to the (same) maximum value
  - Store as a sequence of histograms
    - Also keep list of PRVs of same order
    - Initially empty

$Epid$	$Sick(X)$	$Treat(X,M)$	ϕ_3
$false$	$false$	$false$	$2, ()$
$false$	$false$	$true$	$1, ()$
$false$	$true$	$false$	$2, ()$
$false$	$true$	$true$	$3, ()$
$true$	$false$	$false$	$0, ()$
$true$	$false$	$true$	$1, ()$
$true$	$true$	$false$	$2, ()$
$true$	$true$	$true$	$4, ()$



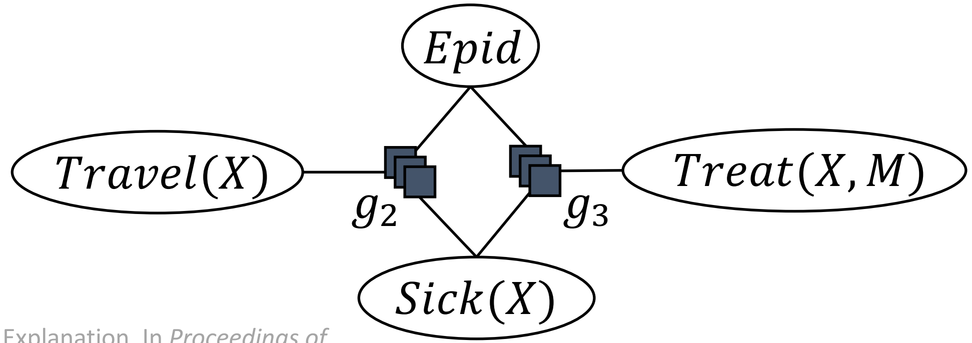
Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

LVE for MPE Queries: Example

- Max-out $Treat(X,M)$
  - Fulfils all preconditions
- Result: $g_3' = \phi_3'(Epid, Sick(X))$
  - Choose argmax assignment for each setting of $Epid, Sick(X)$
    - E.g., $Epid = false, Sick(X) = false$
    - Max. value: 2
    - Assignment leading to 2: $Treat(X,M) = false$
  - New mapping: $(false, false) \rightarrow (2^2, 2 \cdot [0,1])$
    - $\text{ncount}_{M|X}(\top) = 2$

$Epid$	$Sick(X)$	$Treat(X,M)$	ϕ_3
$false$	$false$	$false$	$2,()$
$false$	$false$	$true$	$1,()$
$false$	$true$	$false$	$2,()$
$false$	$true$	$true$	$3,()$
$true$	$false$	$false$	$0,()$
$true$	$false$	$true$	$1,()$
$true$	$true$	$false$	$2,()$
$true$	$true$	$true$	$4,()$

$Epid$	$Sick(X)$	$\phi_3'(.)^{Treat(X,M)}$
$false$	$false$	$2^2, ([0,2])$
$false$	$true$	$3^2, ([2,0])$
$true$	$false$	$1^2, ([2,0])$
$true$	$true$	$4^2, ([2,0])$



Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

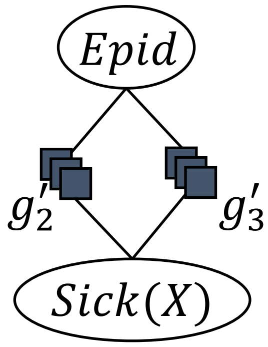
LVE for MPE Queries: Example

- Max-out $Travel(X)$
  - Fulfils all preconditions
  - Result: $g_2' = \phi_2'(Epid, Sick(X))$
- Multiply g_3', g_2'
  - Works as before in terms of multiplication
  - Additionally, concatenate sequences of histograms
    - Cannot contain overlapping assignments due to precondition of max-out
      - Would have been multiplied before

$Epid$	$Sick(X)$	$\phi_3'(.)^{Treat(X,M)}$
$false$	$false$	$2^2, ([0,2])$
$false$	$true$	$3^2, ([2,0])$
$true$	$false$	$1^2, ([2,0])$
$true$	$true$	$4^2, ([2,0])$

$Epid$	$Sick(X)$	$\phi_2'(.)^{Travel(X)}$
$false$	$false$	$4^2, ([1,0])$
$false$	$true$	$2^2, ([1,0])$
$true$	$false$	$5^2, ([0,1])$
$true$	$true$	$7^2, ([0,1])$

$Epid$	$Sick(X)$	$\phi_{23}(.)^{Treat(X,M),Travel(X)}$
$false$	$false$	$2^2 \cdot 4^2, ([0,2],[1,0])$
$false$	$true$	$3^2 \cdot 2^2, ([2,0],[1,0])$
$true$	$false$	$1^2 \cdot 5^2, ([2,0],[0,1])$
$true$	$true$	$4^2 \cdot 7^2, ([2,0],[0,1])$



Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

LVE for MPE Queries: Example

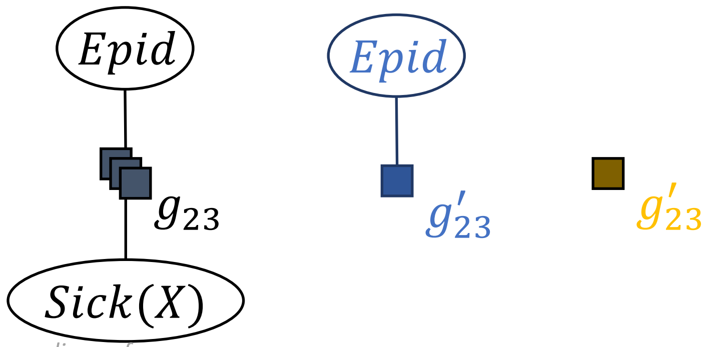
- Max-out $Sick(X)$
  - Fulfils all preconditions
  - Result: $g'_{23} = \phi'_{23}(Epid)$
- Max-out $Epid$
  - Fulfils all preconditions
  - Result: $g''_{23} = \phi''_{23}()$

$Epid$	$\phi''_{23}(.)^{Sick(X),Treat(X,M),Travel(X)}$
$false$	$64, ([0,3], 3 \cdot [0,2], 3 \cdot [1,0])$
$true$	$784, ([3,0], 3 \cdot [2,0], 3 \cdot [0,1])$

$()$	$\phi''_{23}(.)^{Epid,Sick(X),Treat(X,M),Travel(X)}$
	$784, ([1,0], [3,0], [6,0], [0,3])$

$Epid = true$
$\forall X \in \{alice, eve, bob\}, M \in \{m_1, m_2\}:$
$Sick(X) = true,$
$Treat(X, M) = true,$
$Travel(X) = false$

$Epid$	$Sick(X)$	$\phi_{23}(.)^{Treat(X,M),Travel(X)}$
$false$	$false$	$64, ([0,2], [1,0])$
$false$	$true$	$36, ([2,0], [1,0])$
$true$	$false$	$25, ([2,0], [0,1])$
$true$	$true$	$784, ([2,0], [0,1])$



LVE for MPE Queries Continued

- If only PRVs and $\top$ constraints, then maximum assignments will be either all $true$ or all $false$
  - I.e., peak-shaped histograms
- If CRVs in input model, then different histograms might be a maximum assignment
- If inequality constraints in model, then maximum assignment lies where domains split in the middle
  - E.g.,
    - $\phi_1\big(partners(P, C_1, C_2)\big), C_1 \neq C_2$
    - $\phi_2\big(partners(P, C_1, C_2), retail(C_1), retail(C_2)\big), C_1 \neq C_2$
    - Given 15 companies for C_1, C_2, maximum assignment for $retail$ will be at $[8,7]$ and $[7,8]$
      - Independent of the actual potentials, only dependent on the relative max value

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. MPE and Partial Inversion in Lifted Probabilistic Variable Elimination. *AAAI-06 Proceedings of the 21st Conference on Artificial Intelligence*, 2006.

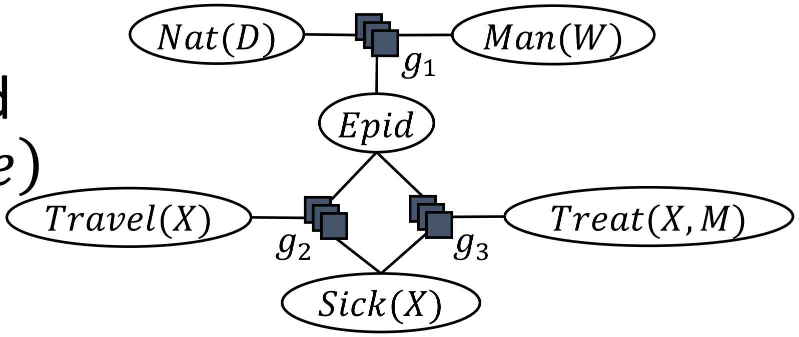
LVE for MPE Queries Continued

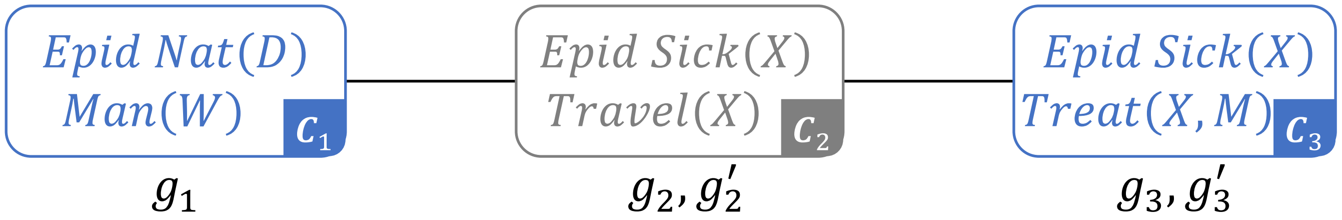
- Count conversions get a bit trickier
  - Logvar that is counted might occur in a maxed out PRV
    - Not a hindrance in terms of counting
    - But: requires counting the logvar in the maxed out PRV as well, requires to rewrite the histograms as well
      - Counting yields histograms $[n, m]$ where the true assignments map to a sequence of histograms and the false assignments to another sequence of histograms
        - The two sequences are basically added up
    - As a consequence:
      Only count a logvar if the PRV turned into a CRV occurs only in the input parfactor
      - If not, multiply together (same as with max-out/sum-out)
      - Why?
        - When a grounding or expansion of the counted logvar would become necessary, it would no longer be possible to trace which grounding leads to which max. value because of the added up histograms
        - Counting only after combining all occurrences prevents a grounding or expansion at a later point (as that can only happen if the original PRV occurs in another parfactor)

Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

LJT for MPE Queries

- (L)JT: message calculation by maxing out
  - Specifically
    - As before: construction, evidence entering
    - Message passing
      - Only **one** message pass (from periphery to centre)
      - Max out remaining variables at centre
    - No explicit query answering step
  - E.g., given model G to right and evidence $sick(alice), sick(eve)$
    - Construct an FO jtree for G
    - Enter $sick(alice), sick(eve)$

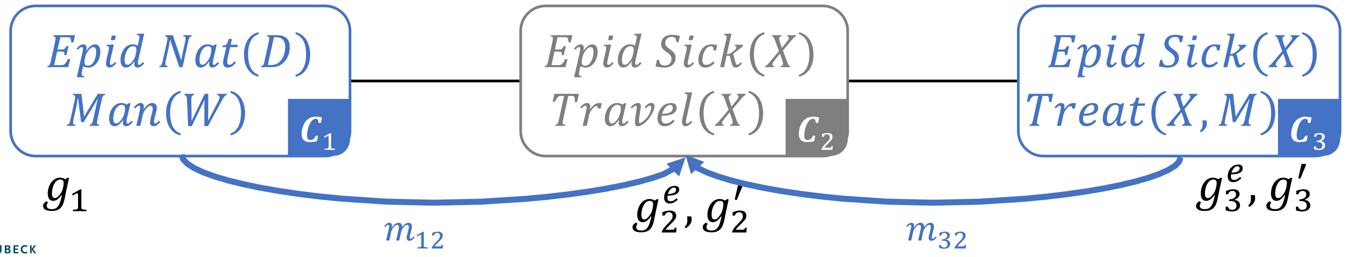




Tanya B and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

LJT: MPE

- Send messages from
  - $\boldsymbol{C}_1$ to $\boldsymbol{C}_2$:
    $m_{12} = \{\phi_1(Epid)^{Nat(D),Man(W)}\}$
  - $\boldsymbol{C}_3$ to $\boldsymbol{C}_2$:
    $m_{32} = \{\phi_3''(Epid, Sick(bob))^{Treat(bob,M)}, \phi_3^{e\prime}(Epid\ )^{Treat(X',M)}\}$
  - At $\boldsymbol{C}_2$:
    - Max-out $Travel(X)$, $Sick(X)$, $Epid$ from G_2, m_{12}, m_{32}
- Result could look like:
  - $Epid = false$
  - $\forall d, w \ \in \top :$ $Nat(d) = false,$ $Man(w) = false$
  - $\forall x' \in \{alice, eve\}, m \in \top :$ $Travel(x') = true,$ $Treat(x', M) = true$
  - $\forall m \in \top : Travel(bob) = true, Sick(bob) = true, Treat(bob, M) = true$



Complexity and Completeness

Complexity

- Runtimes still depend on worst-case parfactor sizes with potentials still being raised to the power of domain sizes
- Results hold from LVE/LJT for probability queries

Completeness

- Maxing out does not affect any argument about completeness of LVE/LJT for
    - $\mathcal{M}^{2lv}$
    - $\mathcal{M}^{1prv}$
- Results hold from LVE/LJT for probability queries (single query terms)

Most Probable Assignment

- *Assignment query* asking for the most probable assignment to a ***subset of randvars*** without evidence: Maximum a posteriori assignment (MAP)
  - Generalisation of MPE
  - Formally, given a model G representing the full joint P_G, evidence $\{E_j = e_j\}_{j=1}^{m}$, $\boldsymbol{e}$ for short, and a set of PRVs $\boldsymbol{U}_{|C'}$,

query terms

$$MAP_G(\boldsymbol{U}_{|C'}|\boldsymbol{e})$$
$$= \operatorname*{argmax}_{\boldsymbol{u} \in \mathcal{R}(\boldsymbol{U}_{|C'})} \sum_{\boldsymbol{t} \in \mathcal{R}(\boldsymbol{T}_{|C''})} P\left((\boldsymbol{U} = \boldsymbol{u})_{|C'}, (\boldsymbol{T} = \boldsymbol{t})_{|C''} | \boldsymbol{e}\right)$$

    - $\boldsymbol{T}_{|C''} = rv(G) \setminus \{E_j\}_{j=1}^{m} \setminus \boldsymbol{U}_{|C'}$ the remaining PRVs
  - If $\boldsymbol{U}_{|C'} = \boldsymbol{V}_{|C}$, i.e., $\boldsymbol{T}_{|C''} = \emptyset$, then $MAP_G(\boldsymbol{U}_{|C'}|\boldsymbol{e}) = MPE_G(\boldsymbol{e})$

Alexander Philip Dawid. Applications of a General Propagation Algorithm for Probabilistic Expert Systems. *Statistics and Computing*, 2(1):25–36, 1992.
Rina Dechter. Bucket Elimination: A Unifying Framework for Probabilistic Inference. In *Learning and Inference in Graphical Models*, pages 75–104. MIT Press, 1999.

Most Probable Assignment

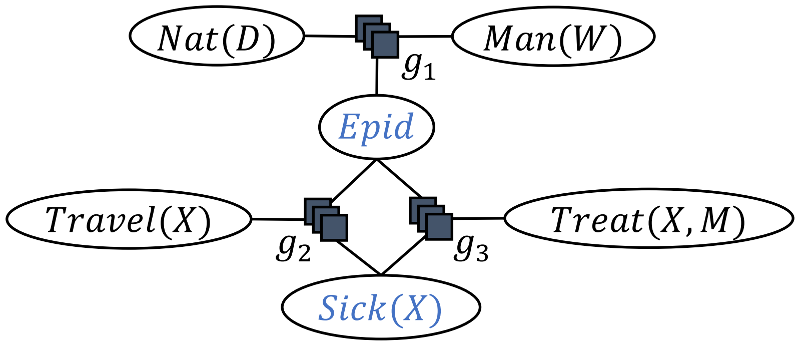
- Problem with an MAP query

$$MAP_G(\boldsymbol{U}_{|C'}|\boldsymbol{e}) = \underset{\boldsymbol{u}\in\mathcal{R}(\boldsymbol{U}_{|C'})}{\operatorname{argmax}} \sum_{\boldsymbol{t}\in\mathcal{R}(\boldsymbol{T}_{|C''})} P\left((\boldsymbol{U}=\boldsymbol{u})_{|C'}, (\boldsymbol{T}=\boldsymbol{t})_{|C''}|\boldsymbol{e}\right)$$

- Contains both summation and maximisation, which are not commutative!
    - One has to
        - first sum out $\boldsymbol{T}_{|C''}$ and
        - only then max out $\boldsymbol{U}_{|C'}$
    - May enlarge tree width
        - Screws with the elimination order

Alexander Philip Dawid. Applications of a General Propagation Algorithm for Probabilistic Expert Systems. *Statistics and Computing*, 2(1):25–36, 1992.
Rina Dechter. Bucket Elimination: A Unifying Framework for Probabilistic Inference. In *Learning and Inference in Graphical Models*, pages 75–104. MIT Press, 1999.

Example



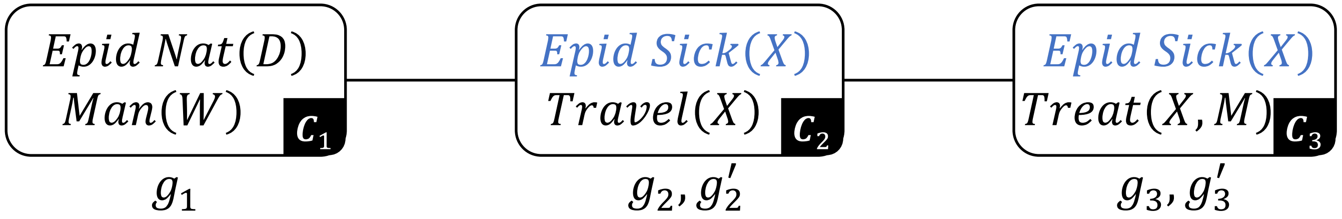
MAP query with same tree width

$$MAP_G\big(Epid, Sick(X)\big)$$

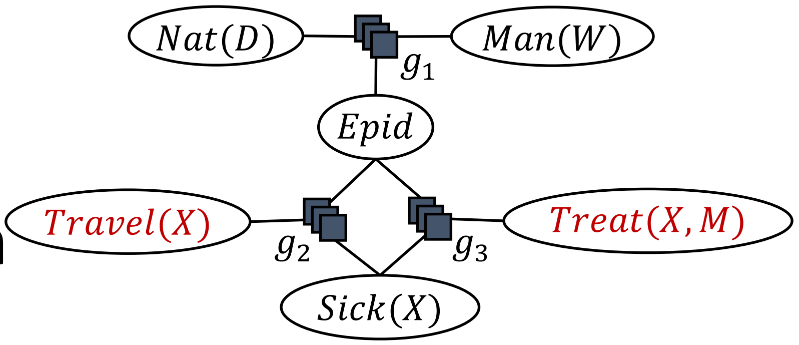
$$= \underset{\substack{s,e\in\\ \mathcal{R}(Epid,Sick(X))}}{\text{argmax}} \sum_{\substack{tl,tt,n,m\in\\ \mathcal{R}\left(\substack{Travel(X),Treat(X,M),\\ Nat(D),Man(W)}\right)}} P(e,s,tl,tt,n,m)$$

$$= \underset{\substack{s,e\in\\ \mathcal{R}(Epid,Sick(X))}}{\text{argmax}} \sum_{\substack{tl,tt,n,m\in\\ \mathcal{R}\left(\substack{Travel(X),Treat(X,M),\\ Nat(D),Man(W)}\right)}} \phi_1(e,n,m)\phi_2(e,s,tl)\phi_3(e,s,tt)$$

$$= \underset{\substack{s,e\in\\ \mathcal{R}(Epid,Sick(X))}}{\text{argmax}} \sum_{\substack{tl\in\\ \mathcal{R}(Travel(X))}} \phi_2(e,s,tl) \sum_{\substack{tt\in\\ \mathcal{R}(Treat(X,M))}} \phi_3(e,s,tt) \sum_{\substack{n,m\in\\ \mathcal{R}(Nat(D),Man(W))}} \phi_1(e,n,m)$$



Example



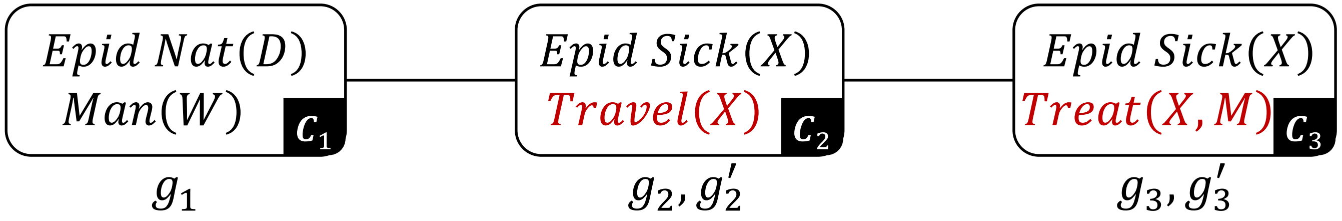
MAP query with enlarged tree width

$$MAP_G(Travel(X),Treat(X,M))$$

$$= \underset{\substack{tl,tt\in\\ \mathcal{R}(Travel(X),Treat(X,M))}}{\operatorname{argmax}} \sum_{\substack{e,s,n,m\in\\ \mathcal{R}\left(\substack{Epid,Sick(X),\\ Nat(D),Man(W)}\right)}} P(e,s,tl,tt,n,m\ )$$

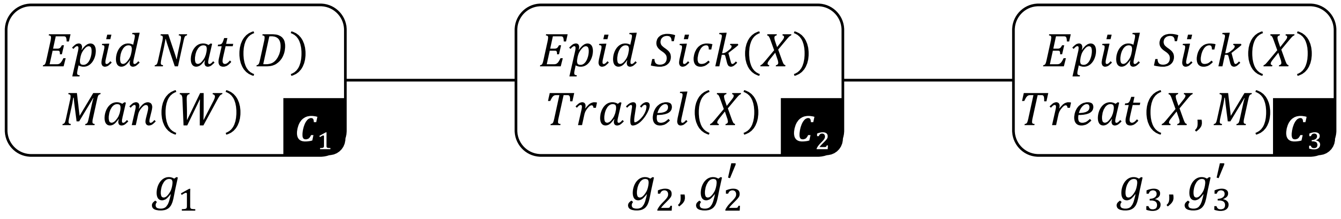
$$= \underset{\substack{tl,tt\in\\ \mathcal{R}(Travel(X),Treat(X,M))}}{\operatorname{argmax}} \sum_{\substack{e,s,n,m\in\\ \mathcal{R}\left(\substack{Epid,Sick(X),\\ Nat(D),Man(W)}\right)}} \phi_1(e,n,m)\phi_2(e,s,tl)\phi_3(e,s,tt)$$

$$= \underset{\substack{tl,tt\in\\ \mathcal{R}(Travel(X),Treat(X,M))}}{\operatorname{argmax}} \sum_{\substack{e\in\\ \mathcal{R}(Epid)}} \sum_{\substack{s\in\\ \mathcal{R}(Sick(X))}} \phi_2(e,s,tl)\phi_3(e,s,tt) \sum_{\substack{n,m\in\\ \mathcal{R}(Nat(D),Man(W))}} \phi_1(e,n,m)$$



Liftable and Bounded MAP Queries

- Liftable MAP queries
  - Does not concern the lifted width, only regards if groundings occur
  - $\mathcal{MAP}^{lift}$: one logvar and one set of constants per query term
    - Argument as with parameterised conjunctive queries regarding completeness
- Bounded MAP queries
  - If query terms cover a subgraph of parclusters, then the lifted width does not increase



LVE for MAP Queries: Algorithm

```
$\mathbf{MAP{-}LVE}$(Model $G$, evidence $\boldsymbol{E}$, query $\boldsymbol{Q}_{|C}$)
    $G \leftarrow$ Shatter $G$ on $\boldsymbol{Q}_{|C}$, $\boldsymbol{E}$, and on itself
    $G \leftarrow$ Absorb $\boldsymbol{E}$ 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 parfactors in $G$
    while $G$ contains query terms do
        if a PRV $A$ fulfils the preconditions of $\mathrm{max{-}out}$ then
            $G \leftarrow$ Apply $\mathrm{max{-}out}$ to $A$ in $G$
        else
            $G \leftarrow$ Apply an enabling operator on parfactors in $G$
    return assignment stored in $G$
```

i.e., **while** $\boldsymbol{T}_{|C''} \neq \emptyset$

Sum-out

Max-out

LJT for MAP Queries: Algorithm

```
$\mathbf{MAP-LJT}(G, \{\boldsymbol{Q}_i\}_{i=1}^n, \boldsymbol{E})$
    Construct an FO jtree $J$ for $G$
    Enter evidence $\boldsymbol{E}$ into $J$
    Pass message in $J$
    **for** each query with query terms $\boldsymbol{Q}_i$ **do**
        Find subgraph $J'$ s.t. $\boxed{\boldsymbol{Q}_i \subseteq rv(J')}$
        Collect submodel $G_{\boldsymbol{Q}}$ from $G_i, m_{ki}, i \in J', k \notin J'$
        Call $\mathrm{MAP-LVE}(G_{\boldsymbol{Q}}, \boldsymbol{Q}, \emptyset)$, return or store result
```

if $\boldsymbol{U}_{|C'} = rv(J')$ **then**
use MPE-LJT to compile
MPE assignment on J'

- Compare query answering to answering for conjunctive queries

MAP Queries Continued

- If given a bounded query, then the complexity results still hold
  - In terms of calculations, combination of LVE and MPE-LVE, which have the same complexity
- Liftability of MAP queries
  - Further work lifting more settings by Sharma et al. (2018)
  - Side note: Sometimes, names are different
    - MPE query = MAP query
    - MAP query = marginal MAP query
    - probability query = marginal query

In logic, corresponding tasks:
⇽ Abduction

⇽ Induction

Vishal Sharma, Noman Ahmed Sheikh, Happy Mittal, Vibhav Gogate, and Parag Singla. Lifted Marginal MAP Inference. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence 4,* 2018.

MAP Queries Continued

Can yield same result but does not have to

- Approximate MAP query $MAP_G(\boldsymbol{U}_{|C'}|\boldsymbol{e})$
  - Replace sum with max
    - Operations for $\boldsymbol{U}_{|C'}$ and $\boldsymbol{T}_{|C''}$ are commutative again
    - Basically computes an MPE query with the result projected onto the query terms of the MAP query
    - Provides a lower bound in terms of the max potential at the end

$$\widetilde{MAP}_G(\boldsymbol{U}_{|C'}|\boldsymbol{e}) = \operatorname*{argmax}_{\boldsymbol{u}\in\mathcal{R}(\boldsymbol{U}_{|C'})} \max_{\boldsymbol{t}\in\mathcal{R}(\boldsymbol{T}_{|C''})} P\left((\boldsymbol{U}=\boldsymbol{u})_{|C'},(\boldsymbol{T}=\boldsymbol{t})_{|C''}|\boldsymbol{e}\right)$$

$$= \pi_{\boldsymbol{U}_{|C'}}(MPE_G(\boldsymbol{e}))$$

A	B	ϕ	$\sum_B$	argmax_A
$false$	$false$	1	6 ^{4}	
$false$	$true$	5 ^{3}		
$true$	$false$	3	7	$A = true$
$true$	$true$	4		

$MAP_\phi(A)$

A	B	ϕ	$\max_B$	argmax_A
$false$	$false$	1	5 ^{3}	$A = false$
$false$	$true$	5 ^{3}		
$true$	$false$	3	4	$A = true$
$true$	$true$	4		

$\widetilde{MAP}_\phi(A)$

A Combined LJT for All Query Types

- Given queries of different types
  - Types:
    - MPE
    - MAP
    - Probability
      - Parameterised, conjunctive query for a (conditional) probability (distribution)
- On one evidence set
- Can reuse FO jtree at different stages

```
$\mathbf{ComLJT}(G, \{\boldsymbol{Q}_i\}_{i=1}^{n}, \boldsymbol{E})$
    Construct an FO jtree $J$ for $G$
    Enter evidence $\boldsymbol{E}$ into $J$
    Pass message in $J$
    **for** each query with query terms $\boldsymbol{Q}_{|C'}$ and a type **do**
        **if** MPE query **then**
            Pass MPE messages using $J$
        **else**
            Find subgraph $J'$ s.t. $\boldsymbol{Q}_{|C'} \subseteq rv(J')$
            **if** MAP query, $\boldsymbol{Q}_{|C'} = rv(J')$ **then**
                Pass MPE messages using $J'$
            **else**
                Extract a submodel $G'$
                **if** MAP query **then**
                    $\mathrm{MAP{-}LVE}\left(G', \boldsymbol{Q}_{|C'}, \emptyset\right)$
                **else**
                    $\mathrm{LVE}\left(G', \boldsymbol{Q}_{|C'}, \emptyset\right)$
```

Interim Summary

- Adaptive inference
  - FO jtree structure remains valid: Adaptive message passing
  - Adapt structure: Reasonable with locally restricted changes
- Leaving a specific domain behind
  - Changing domains
    - Without effect on query results:
      (conditional) independence, projectivity
    - With effect: Adapt weights
  - Unknown domains
    - Sets of or distributions over models
    - New query types
- Most probable assignments
  - Replace $\sum$ by argmax for assignment PRVs
    - Problem if both $\sum$ and argmax occur: not commutative!
    - Identical lifted width if assignment on whole parclusters

Outline: 3. Lifted Inference

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

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

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

D. *Beyond Standard Query Answering*
   - Adaptive inference
   - Changing and unknown domains
   - Assignment queries

$\Longrightarrow$ Next: Lifted learning