Intelligent Agents:
Web-mining Agents

Probabilistic Graphical Models

Sequential Models & Inference

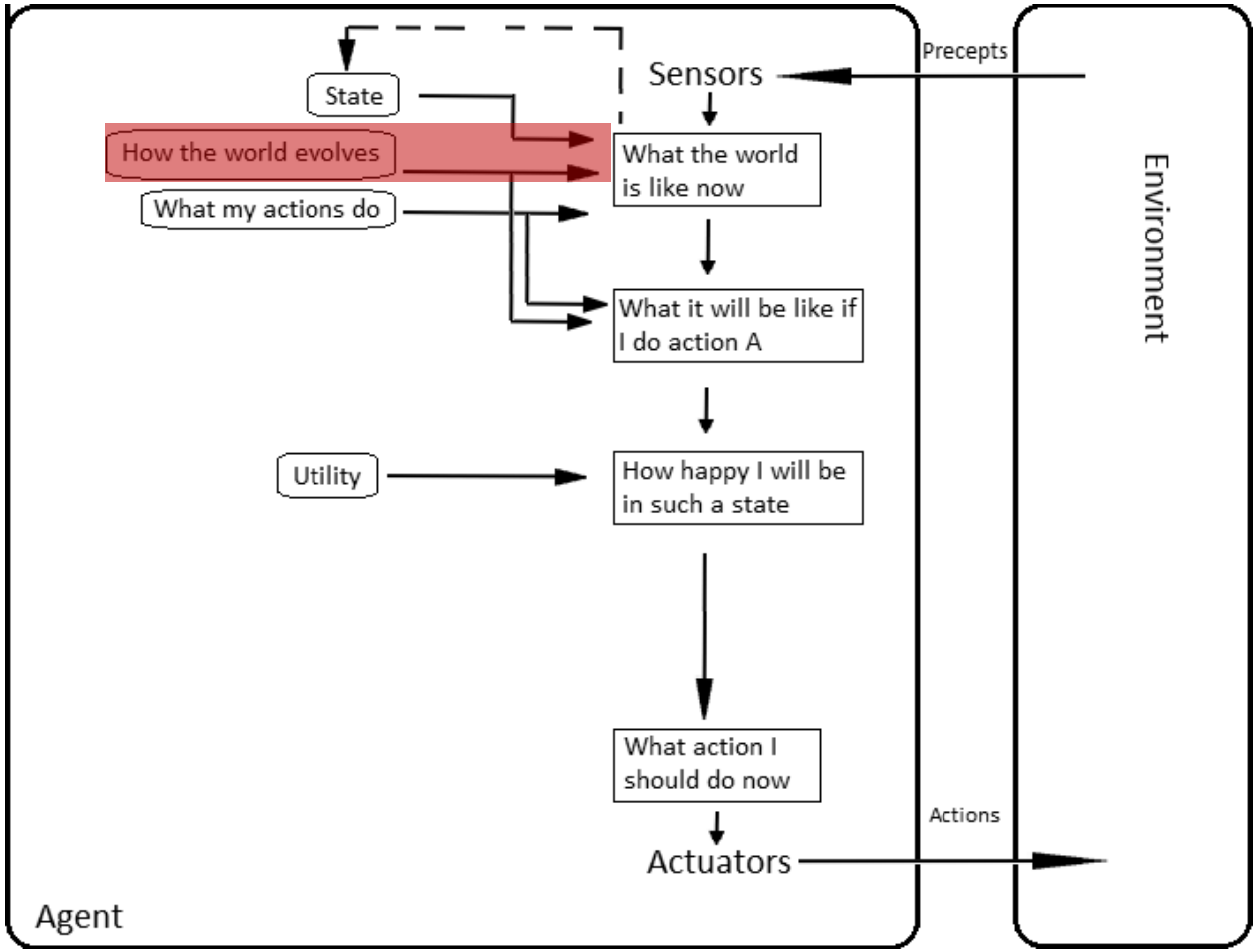
Tanya Braun

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

Setting: Agent with Utilities



A Note on Naming Conventions

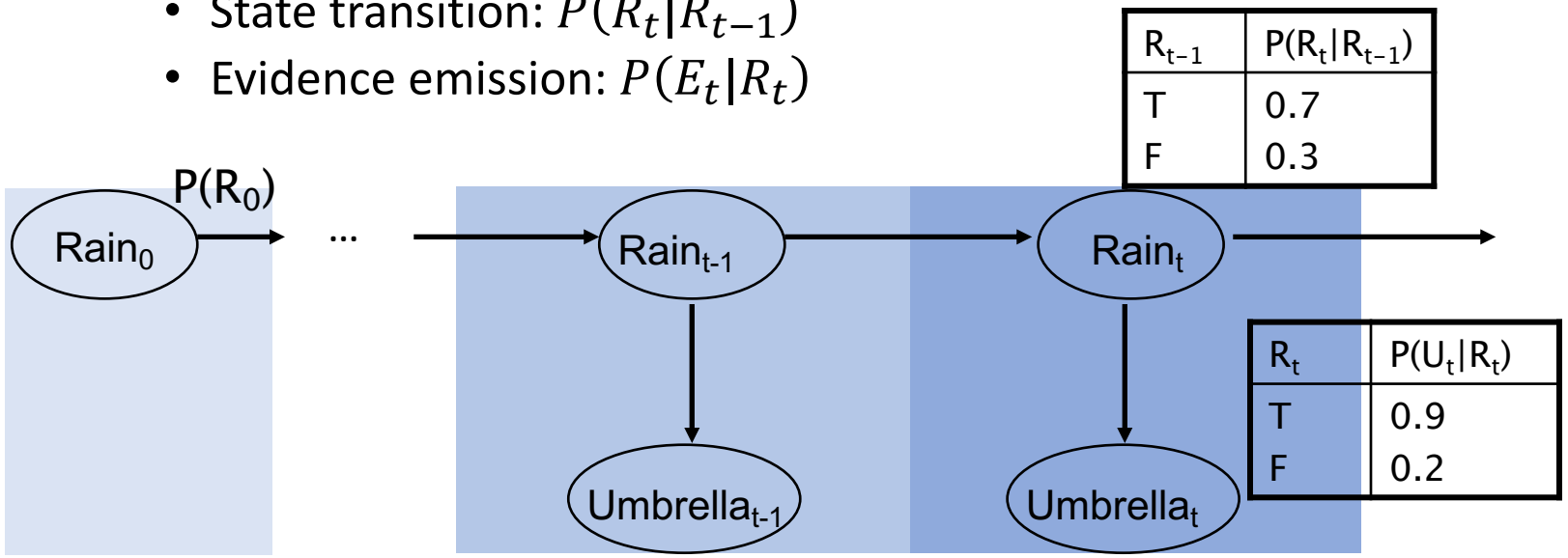
- Common names for the same thing in PGMs
  - Dynamic
    - BUT: *stationary* in terms of how a state changes from the previous the current one
      - State does change and has an influence on the next one
  - Temporal
    - Changes between states considered due to time moving forward, i.e., a temporal state sequence
      - Implicit direction of edges towards the future
      - Simplifying assumption: Discrete time steps indexed by integer (t)
  - Sequential
    - Generalised version of the notion “temporal” as the sequence may occur not only due to time moving forward but because of something else (e.g., spatial movement, sequence of words in text; implicitly, then also time moves forward)

Remember in IR Topic 4: DBNs

- Actually, a specific DBN: Hidden Markov model (HMM)
  - One state randvar R_t
    - Latent (hidden)
  - One evidence randvar E_t
    - Observable

  → Two CPTs (+ prior for R_0)
    - State transition: $P(R_t|R_{t-1})$
    - Evidence emission: $P(E_t|R_t)$

- Copy pattern over t with a start description for $t = 0$

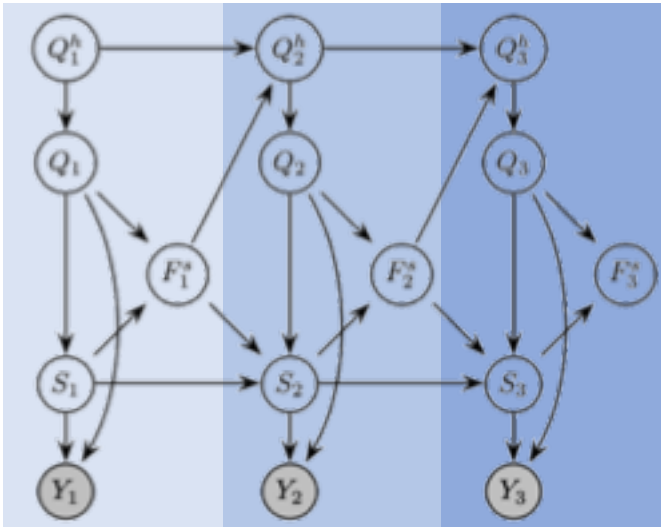


R_{t-1}	$P(R_t\|R_{t-1})$
T	0.7
F	0.3

R_t	$P(U_t\|R_t)$
T	0.9
F	0.2

DBNs: Generalising HMMs

- Set of randvars
  - Some latent, some observable (marked grey in figure)
- Set of CPTs connecting randvars (+ description for $t = 0$)
  - Within a time step: only t occur as indices
  - Between time steps: different t's occur as indices
    - If only t and $t-1$ occur: Markov-1 assumption (as in HMMs)



Time slices

- Copied for each time step
- Turn a discrete DBN into an HMM by combining all latent randvars into one and all observable randvars into another randvar, multiplying CPTs accordingly
  - Gives up the (conditional) independences between randvars → may (greatly) increase the number of entries in the CPTs

Figure: Kevin P. Murphy: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley, 2002.

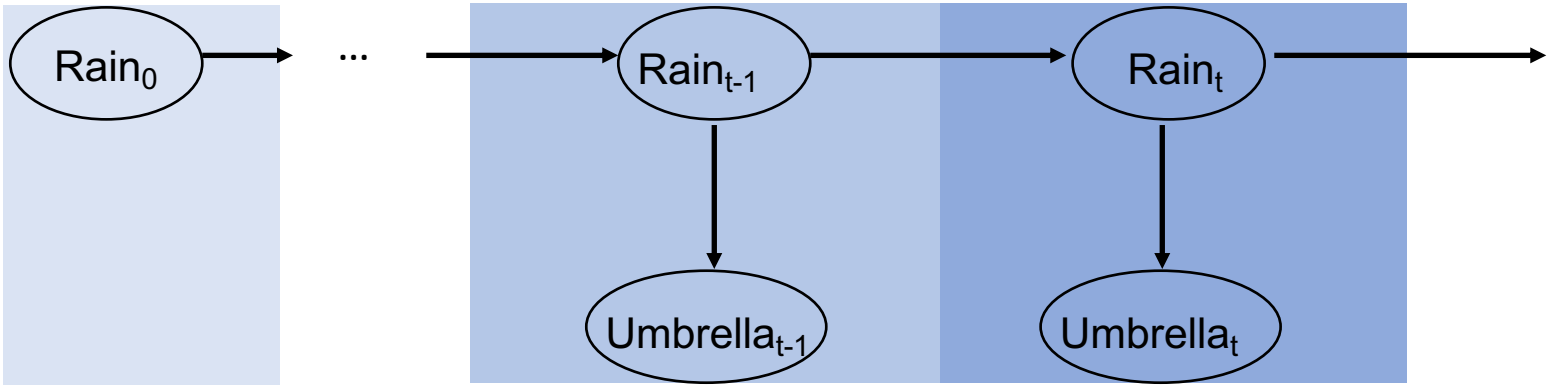
Tasks in HMMs

- Query $P(R_\pi | E_{0:\tau})$, τ the current step
  - Filtering: $\pi = \tau$
  - Prediction: $\pi > \tau$
  - Hindsight: $\pi < \tau$
    - Also called smoothing
  - MPE
    - In HMMs: solved by Viterbi algorithm

Shorthand notation for a set of observations for E over all steps from 0 to t

- State variable is a separating subset between the past, i.e., all randvars indexed with $t < \tau$, and the future, i.e., all randvars indexed with $t > \tau$

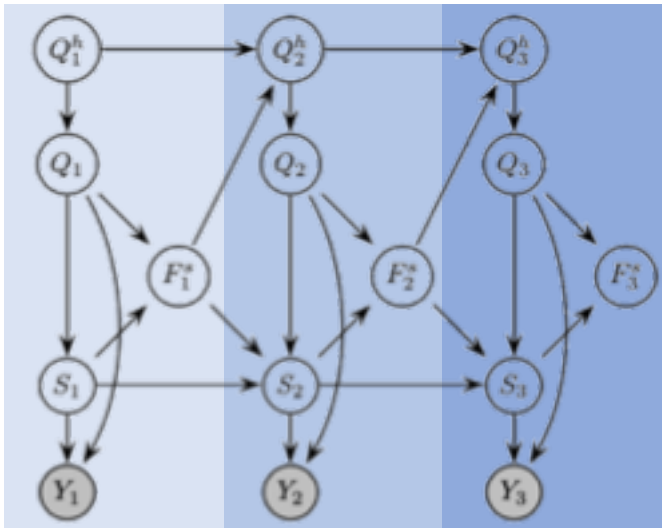
→ Allows for propagation algorithm
  - Forward pass for filtering/prediction
  - Additional backward pass for hindsight



Time slices

Solving Tasks in DBNs

- Not one randvar that separates past from future
- Find separating subset that make the past independent from the present and the present independent from the future
  - So called interface
    - Separates current slice from previous and subsequent slices
- To automatically find interfaces and specify a similar propagation algorithm, we will use jtrees again
  - Collect information about past at forward interface and send it onwards



Time slices

Kevin P. Murphy: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley, 2002.

Outline: 6. Sequential Models & Inf.

A. Lifted modelling of sequences

- Parameterised dynamic models (PDMs)
- Modelling, semantics

B. Lifted dynamic inference

- Inference tasks
- Interfaces
- Lifted dynamic junction tree algorithm (LDJT)
- Theoretical analysis: complexity

C. Keeping inference polynomial

- Problem of evidence over time in lifted models
- Temporal approximate merging (TAMe)

Step/Time-indexed PRVs

- PRVs get an index referring to its position in the sequence of states/time
  - Previous version: $A = R(X_1, \dots, X_n)$
    - Combination of a randvar name R and n logvars X_i
      - If $n = 0$: $A = R$ constitutes a propositional randvar
  - Sequential version: $A_t = R(X^1, \dots, X^n)_t$
    - *Sequential indices as subscript, other indices as superscript*
    - Combination of a randvar name R and n logvars X_i and an index t
      - If $n = 0$: $A_t = R_t$ constitutes a propositional randvar indexed by t
    - Only the PRV as a whole is indexed by t, not the individual logvars
      - I.e., $R(X_1, \dots, X_n)_t \neq R(X_1, \dots, X_n)_{t-1}$
      - But, $lv(R(X_1, \dots, X_n)_t) = \{X_1, \dots, X_n\} = lv(R(X_1, \dots, X_n)_{t-1})$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Parameterised Dynamic Models (PDMs)

- Assumptions: Markov-1, stationary process
- PDM $G = (G_0, G_{\rightarrow})$ where
  - G_0 is a PM describing the intra-time slice behaviour for $t = 0$:

$$G_0 = \{g_0^i\}_{i=1}^{n_0}$$

    - $g_0^i = \phi_0^i\left(R_0^1, \ldots, R_0^{l_i}\right)_{|C_0^i}$
  - $G_{\rightarrow}$ is a PM describing the intra- and inter-time slice behaviour for $t > 0$

$$G_{\rightarrow} = G_{t-1} \cup G_t \cup G_{t-1,t}$$

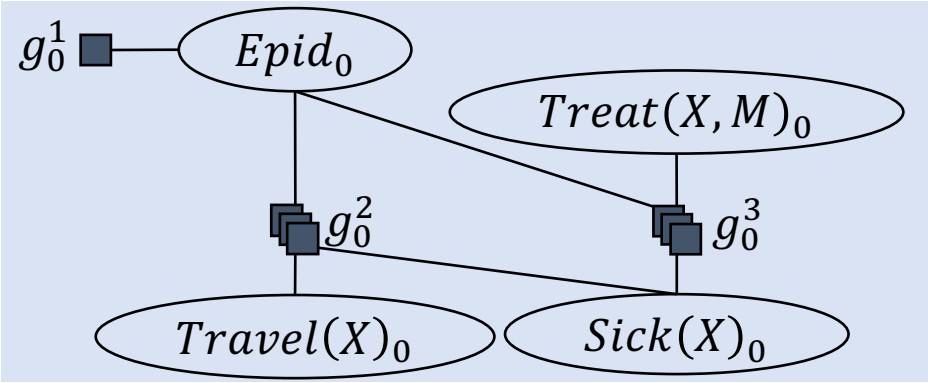
    - $G_t = \{g_t^k\}_{k=1}^{n_t}, g_t^k = \phi_t^k\left(A_t^1, \ldots, A_t^{l_k}\right)_{|C_t^k}$
    - $G_{t-1} = G_{t|t\ replaced\ by\ t-1}$
    - $G_{t-1,t} = \{g^j\}_{j=1}^{n}, g^j = \phi\left(A_\pi^1, \ldots, A_\pi^{l_j}\right)_{|C}, \pi \in \{t-1, t\}$
  - In the general setting, it usually holds $G_t \subseteq G_{0|0\ replaced\ by\ t}$
    - i.e., there is a correspondence between G_0 and G_t (and G_{t-1})

Also called a 2-(time) slice model as it contains descriptions for two time slices

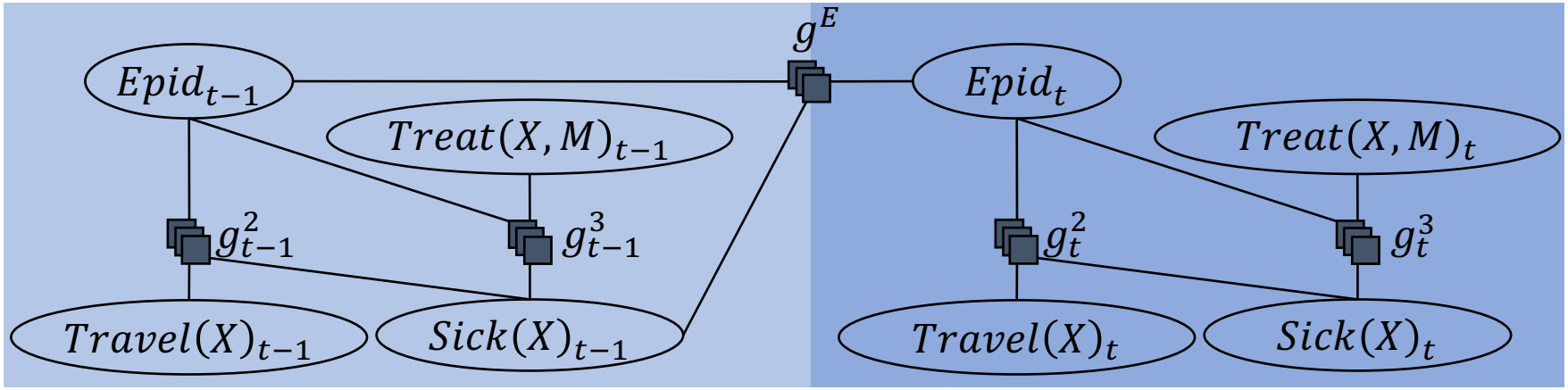
Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

PDM: Example

- $G_0 = \{g_0^1, g_0^2, g_0^3\}$



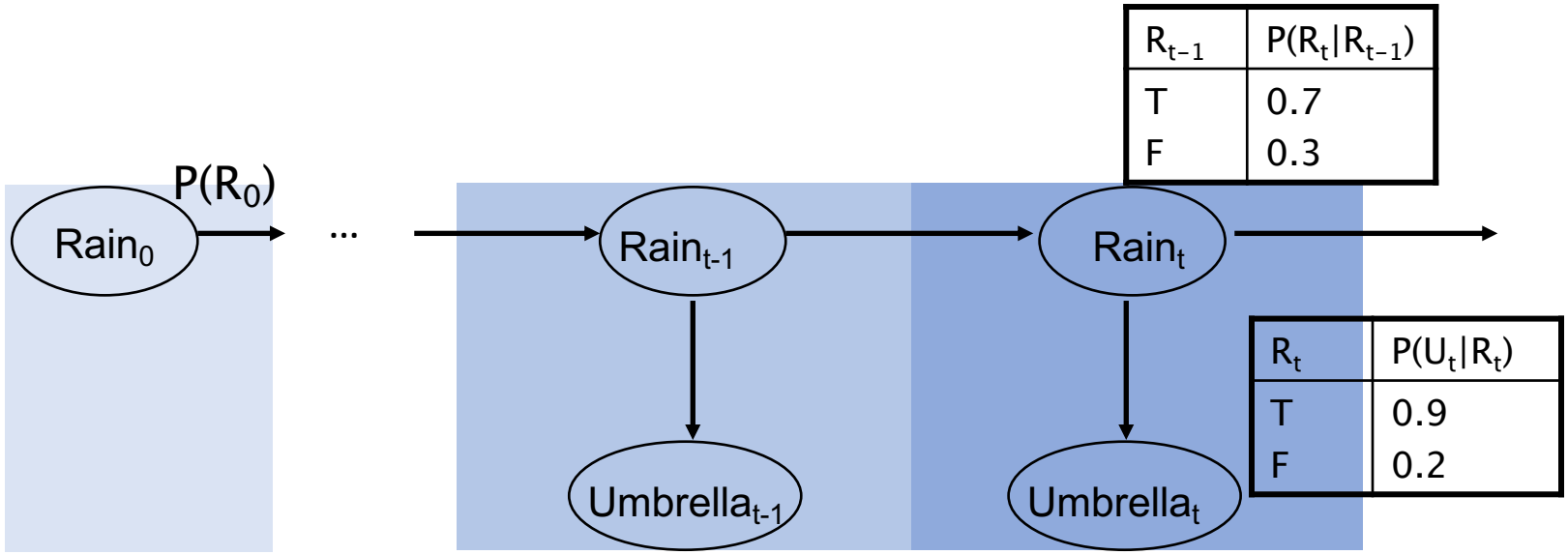
- $G_{\rightarrow} = \{g_{t-1}^2, g_{t-1}^3\} \cup \{g_t^2, g_t^3\} \cup \{g^E\}$



Time slices

HMMs as PDMs

- PDM $G = (G_0, G_{\rightarrow})$
  - $G_0 = \{g^i\}_{i=1}^{n_0}$
  - $G_{\rightarrow} = G_{t-1} \cup G_t \cup G_{t-1,t}$
- In the HMM setting,
  - $G_0 = \{P(R_0)\}$
  - $G_t = \{P(E_t|R_t)\}$
  - $G_{t-1} = \{P(E_{t-1}|R_{t-1})\}$
  - $G_{t-1,t} = \{P(R_t|R_{t-1})\}$



Time slices

1.5-slice Model

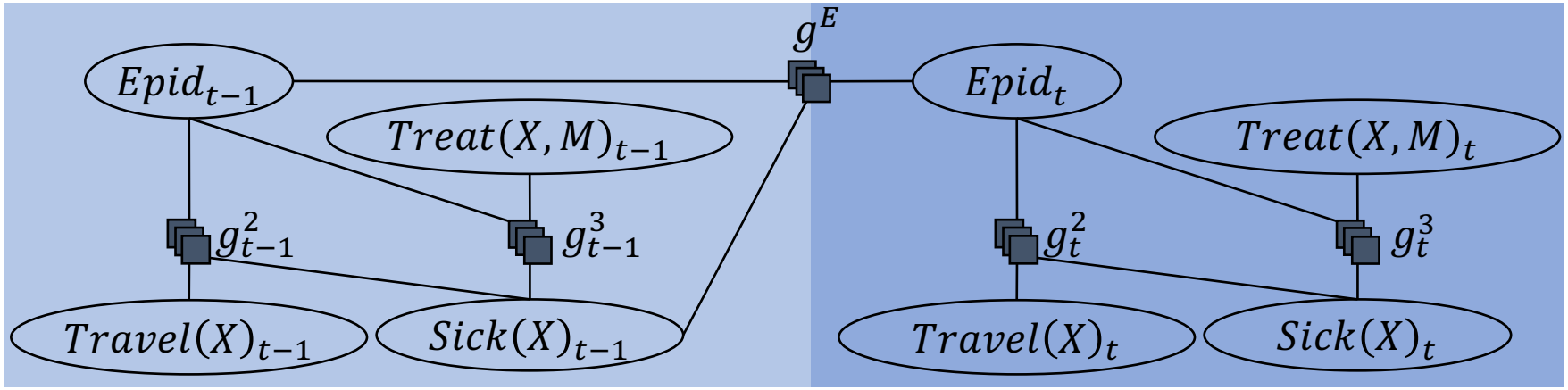
- Information about transition behaviour and one (time) slice
  - Defines a copy pattern for each new step appended
- Formally,

$$G_{\rightarrow}^{1.5} = G_t \cup G_{t-1,t}$$

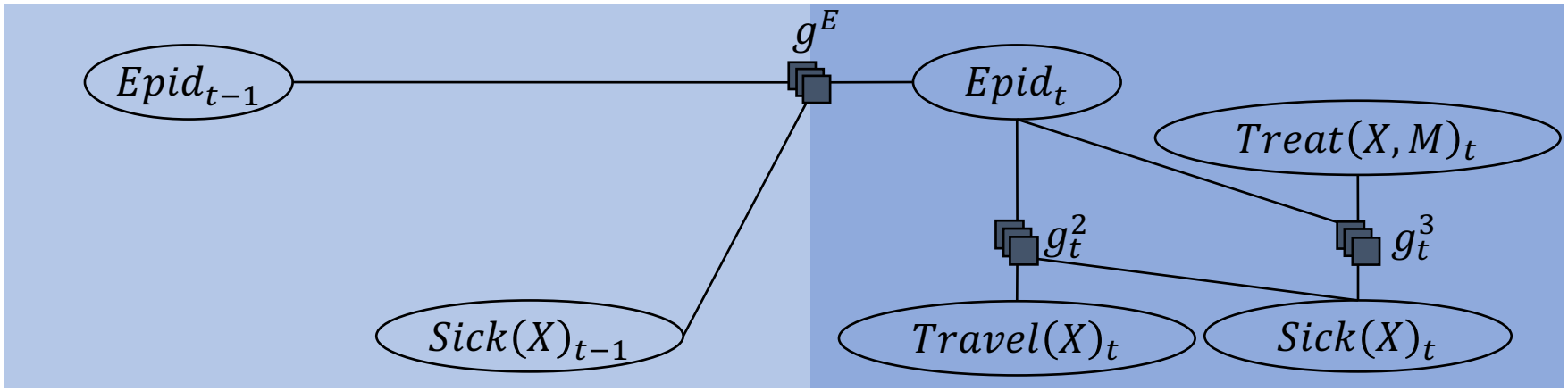
  - where G_t and $G_{t-1,t}$ are defined as before
    - $G_t = \{g_t^k\}_{k=1}^{n_t}, g_t^k = \phi_t^k(A_t^1, \ldots, A_t^{l_k})_{|C_t^k}$
    - $G_{t-1,t} = \{g^j\}_{j=1}^{n}, g^j = \phi\left(A_\pi^1, \ldots, A_\pi^{l_j}\right)_{|C}, \pi \in \{t-1, t\}$
- If given a 2-slice model $G_{\rightarrow}$, remove all parfactors g where $rv(g)$ contains only PRVs with index $t-1$

1.5-slice Model: Example

- 2-slice model



- 1.5-slice model



Unrolling a PDM

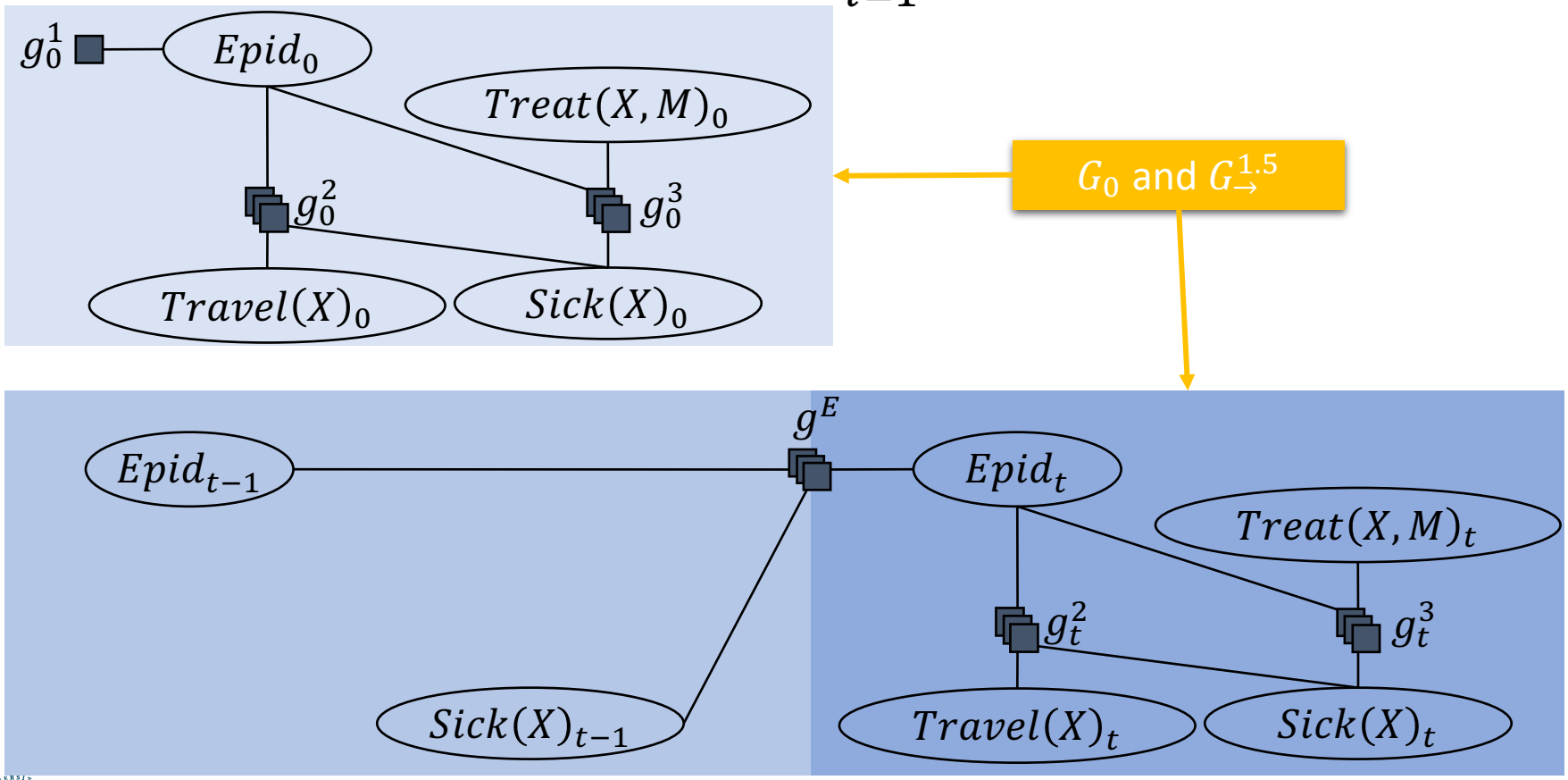
- Given a maximum step size T, unroll G for T steps
  - Start with $\tau = 0$: Use G_0
  - For each $\tau < T$: Instantiate $G_{\rightarrow}^{1.5}$ for τ
    - Instantiate: replace t by τ
    - PRVs with index $\tau - 1$ refer to PRVs instantiated for $\tau - 1$
      - Especially obvious in a graphical representation
- Formally, given a PDM $G = (G_0, G_{\rightarrow})$ and number T

$$G_{0:T} = unroll(G, T) = G_0 \cup \bigcup_{\tau=1}^{T} G_{\rightarrow|t\ replaced\ by\ \tau}^{1.5}$$

  - $G_{0:T}$ is a standard PM as defined before

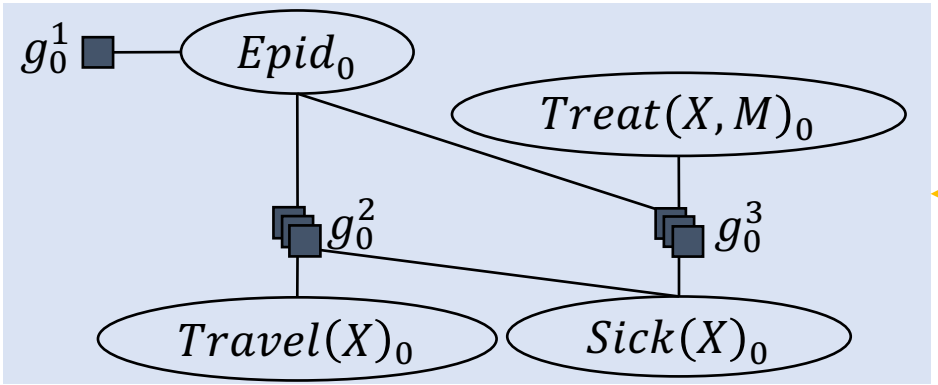
Unrolling a PDM: Example – $T = 2$

$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \; replaced \; by \; \tau}$$



Unrolling a PDM: Example – $\tau = 0$

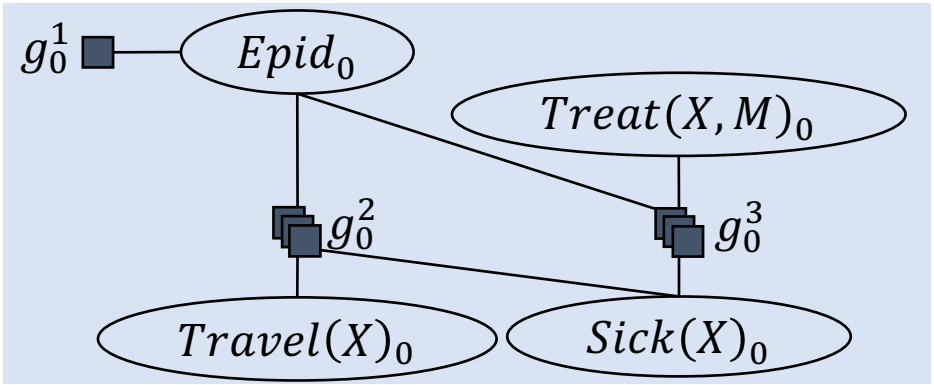
$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \; replaced \; by \; \tau}$$



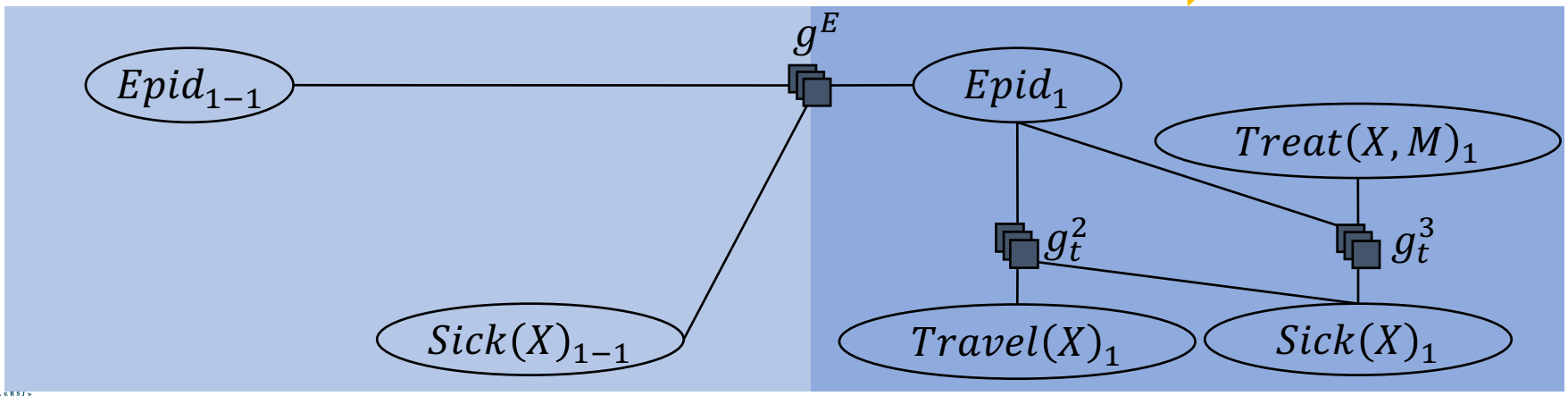
As $\tau = 0$, use G_0

Unrolling a PDM: Example – $\tau = 1$

$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \text{ replaced by } \tau}$$

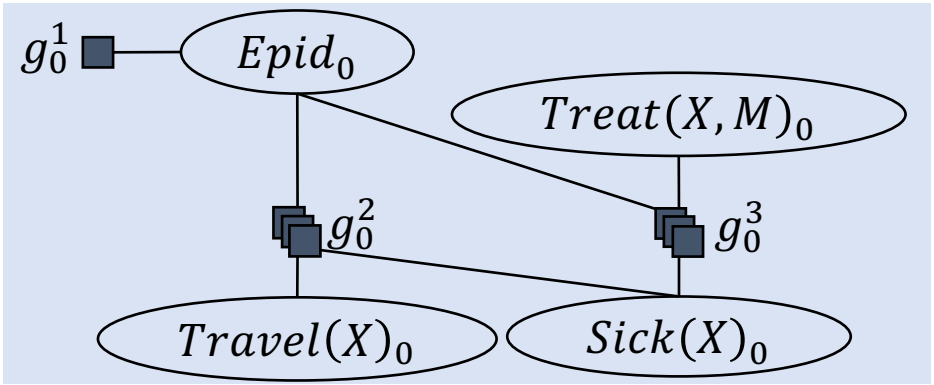


Instantiate $G^{1.5}_{\rightarrow}$ for $\tau = 1$, i.e., replace t with τ

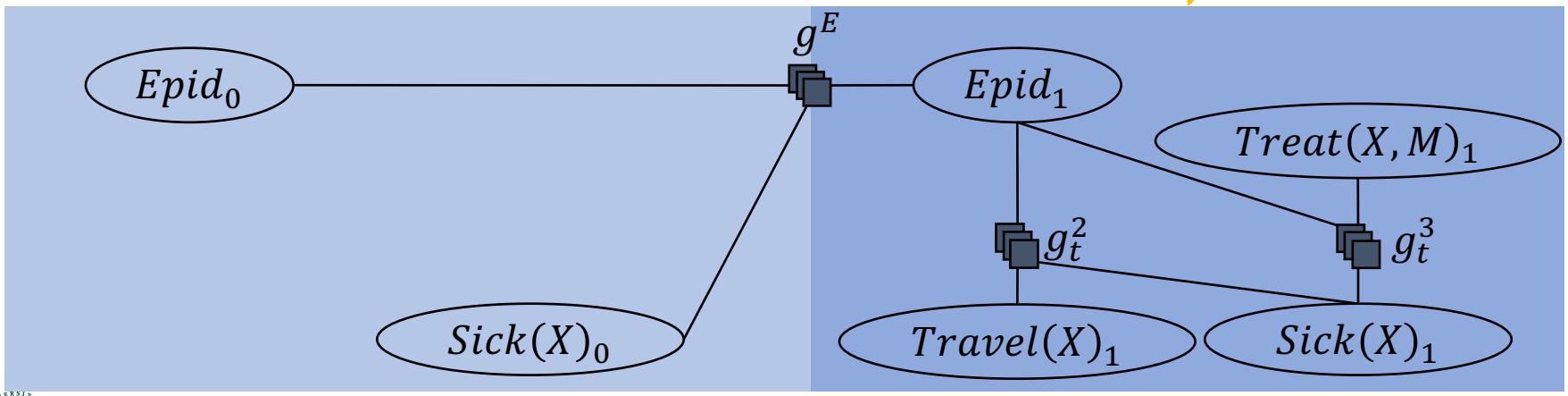


Unrolling a PDM: Example – $\tau = 1$

$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \; replaced \; by \; \tau}$$



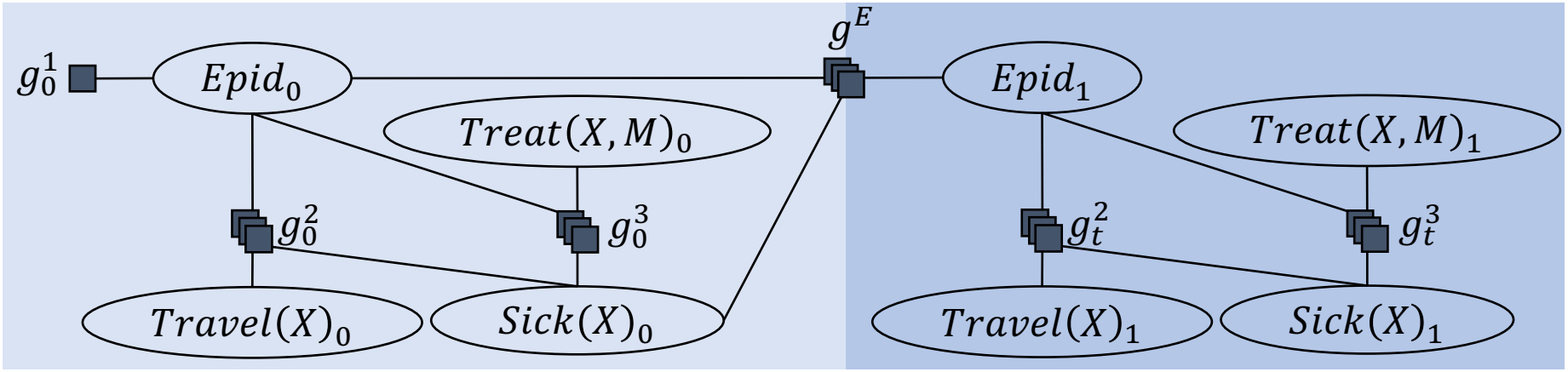
Add to G_0 (append to G_0 in terms of the graph)



Unrolling a PDM: Example – $\tau = 1$

$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \text{ replaced by } \tau}$$

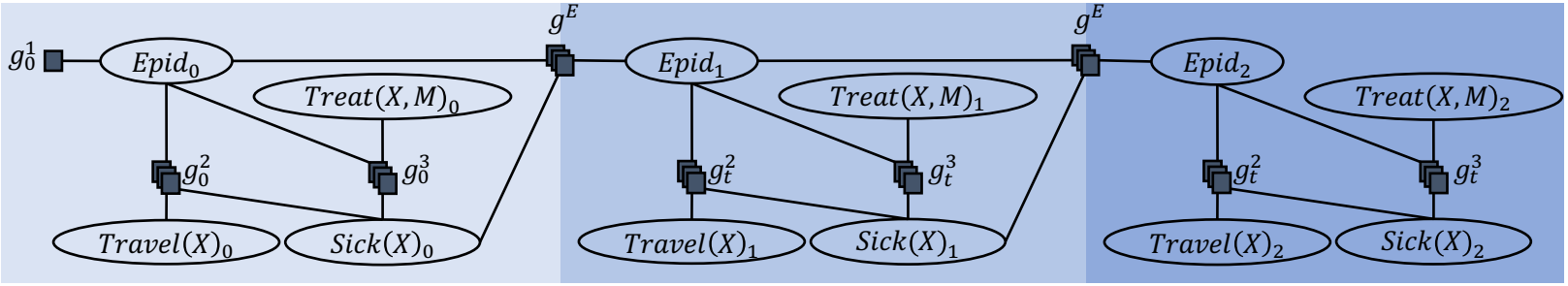
$$G_{0:\tau} = G_{0:1} = G_0 \cup G^{1.5}_{\rightarrow | t \text{ replaced by } 1}$$



Unrolling a PDM: Example – $\tau = 2$

$$G_{0:2} = unroll(G, 2) = G_0 \cup \bigcup_{\tau=1}^{2} G^{1.5}_{\rightarrow | t \text{ replaced by } \tau}$$

Instantiate $G^{1.5}_{\rightarrow}$ for $\tau = 2$ and add to $G_{0:1}$



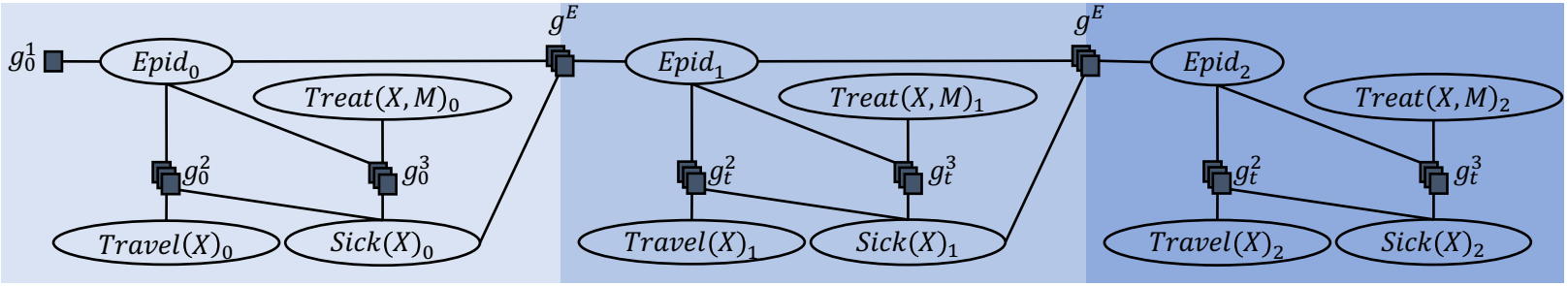
$$G_{0:\tau} = G_{0:2} = unroll(G, 2)$$

PDM: Semantics

- Given a PDM $G = (G_0, G_{\rightarrow})$ and number T, semantics as defined for PMs
  - Unrolling for T steps, grounding, and building a full joint distribution

$$P_{(G,T)} = \frac{1}{Z} \prod_{f \in gr(G_{0:T})} f$$

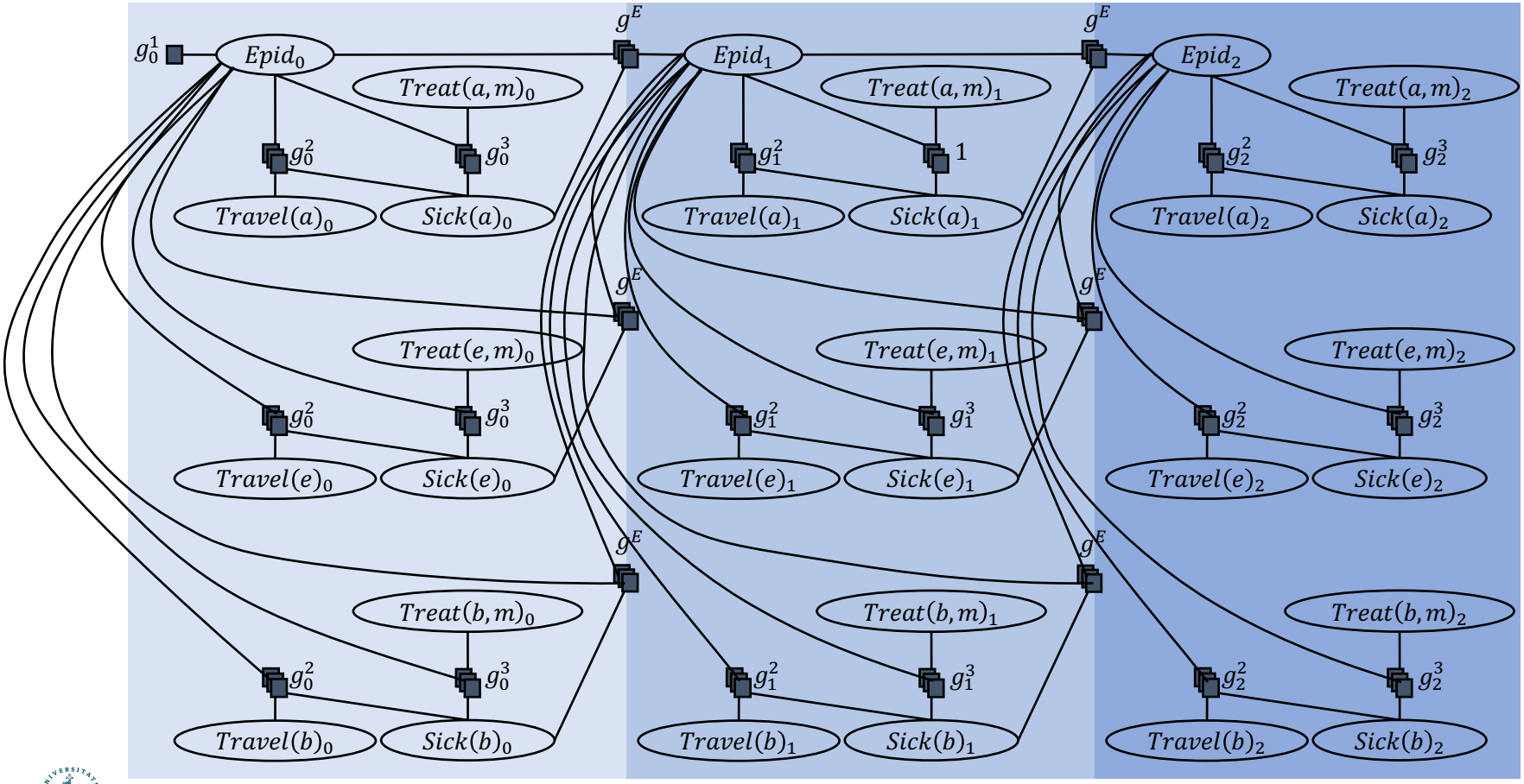
- After unrolling for $T = 2$



Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

PDM: Semantics

- After grounding
  - $\mathcal{D}(X) = \{a, e, b\}, \mathcal{D}(M) = \{m\}$



PDM: Semantics

- Full joint is then a probability distribution over

$$Epid_0, Epid_1, Epid_2$$
$$Sick(a)_0, Sick(a)_1, Sick(a)_2$$
$$Sick(e)_0, Sick(e)_1, Sick(e)_2$$
$$Sick(b)_0, Sick(b)_1, Sick(b)_2$$
$$Travel(a)_0, Travel(a)_1, Travel(a)_2$$
$$Travel(e)_0, Travel(e)_1, Travel(e)_2$$
$$Travel(b)_0, Travel(b)_1, Travel(b)_2$$
$$Treat(a,m)_0, Treat(a,m)_1, Treat(a,m)_2$$
$$Treat(e,m)_0, Treat(e,m)_1, Treat(e,m)_2$$
$$Treat(b,m)_0, Treat(b,m)_1, Treat(b,m)_2$$

- Size: 2^{30}
  - Boolean ranges

Interim Summary

- Modelling Sequential Data
  - Assumption: Markov-1, stationary process
  - Dynamic model consists of two static models
    - One to describe the first step
    - One to describe the transition from one to the other
      - Copy pattern
  - Semantics by unrolling for T steps

- Actually, whether it is HMMs, DBNs, PDMs or *dynamic MLNs* → same basic structure/idea

Outline: 6. Sequential Models & Inf.

A. *Lifted modelling of sequences*
   - Parameterised dynamic models (PDMs)
   - Modelling, semantics

B. *Lifted dynamic inference*
   - Inference tasks
   - Interfaces
   - Lifted dynamic junction tree algorithm (LDJT)
   - Theoretical analysis: complexity, completeness

C. *Keeping inference polynomial*
   - Problem of evidence over time in lifted models
   - Temporal approximate merging (TAMe)

PDM: Tasks

- As before: Query $P(R_\pi | \boldsymbol{E}_{0:\tau})$, τ the current step
  - Filtering: $\pi = \tau$
  - Prediction: $\pi > \tau$
  - Hindsight (smoothing): $\pi < \tau$
- Restricted to one (grounded) PRV or propositional randvar R_π as a query term for the moment
  - Parameterised queries ok if single query term
- $\boldsymbol{E}_{0:\tau}$ contains for each step τ' a set of events

$$\left\{ \left\{ E^i_{\tau'} = e \right\}_{i=1}^{n} \right\}_{\tau'=0}^{\tau}$$

  - $e \in \mathcal{R}(E^i_t)$

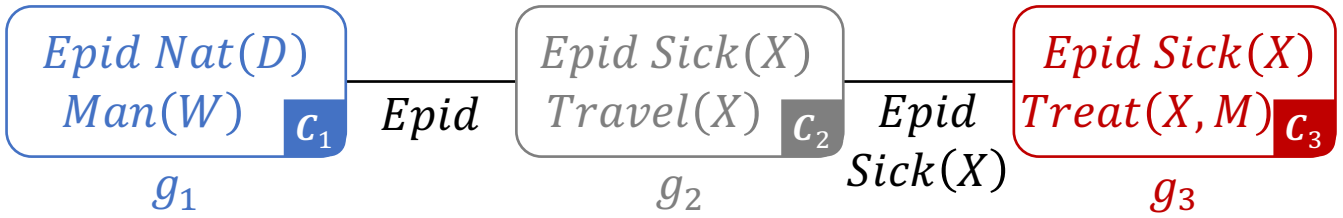
Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Naïve Inference by Unrolling

- Given a PDM $G = (G_0, G_\rightarrow)$, (a number T), and a query $P(R_\pi | \boldsymbol{E}_{0:T'})$
  - Unroll model for T steps and use any inference algorithm of one's liking to answer $P(R_\pi | \boldsymbol{E}_{0:T'})$
    - $T' \leq T, \pi \leq T$, could determine T to be $T = \max\{T', \pi\}$
- Problems:
  - Unrolled models get very large
  - Restart if T, R_π, $\boldsymbol{E}_{0:T'}$ changes
    - Scenario of step τ increasing with new evidence coming in
      - New $G_\rightarrow^{1.5}$ has to be added, evidence re-entered/added
- Naïve inference on unrolled models not efficient
- Aim: Work with one current model that can efficiently handle increasing τ and different queries

Clustering

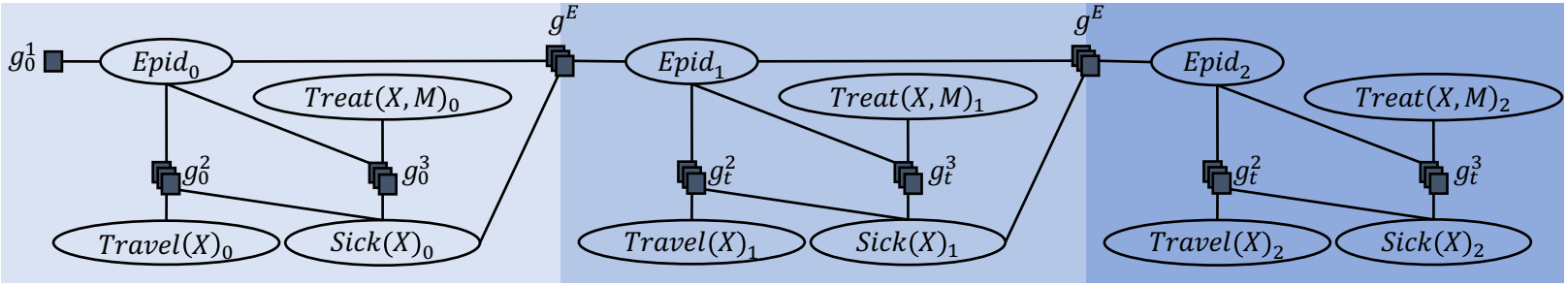
- Remember: Parclusters are sets of PRVs enough for query answering
  - Arranged in an acyclic graph (FO jtree)
  - Messages provide information over separators of remaining part of model, make a parcluster independent from its neighbours



- Want something similar for sequential inference where slices are independent given separating subsets, so-called interfaces, between them

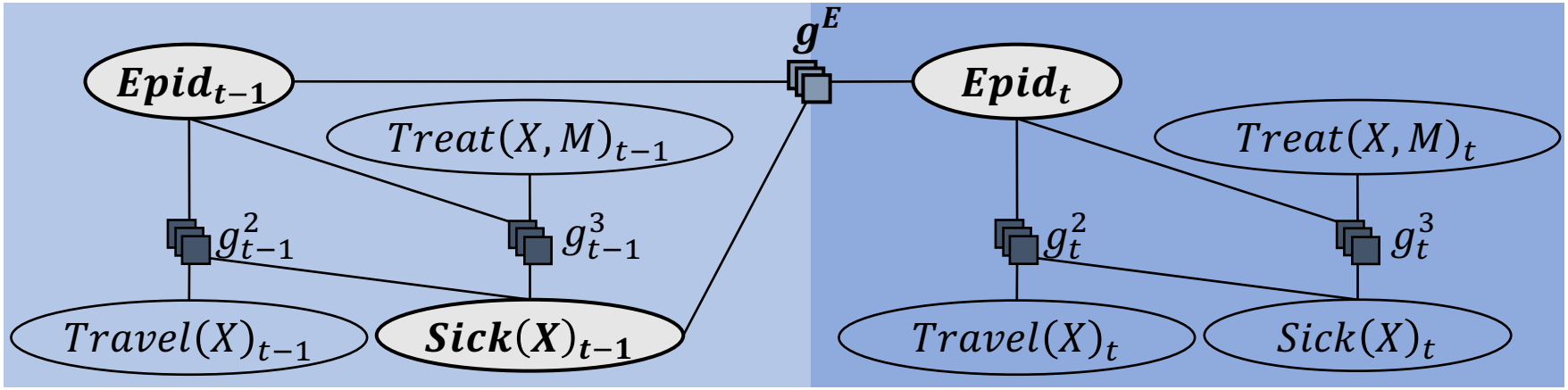
Clustering

- In unrolled example, want interfaces such that the following independences hold
  - G_0 independent from $G_{1:2}$
  - G_1 independent from G_0, G_2
  - G_2 independent from $G_{0:1}$



PDM: Interfaces

- Separating subsets that make the past independent from the present and the present independent from the future
  - Relevant model parts: transition parfactors
    - Those that contain PRVs with index $t-1$ and t



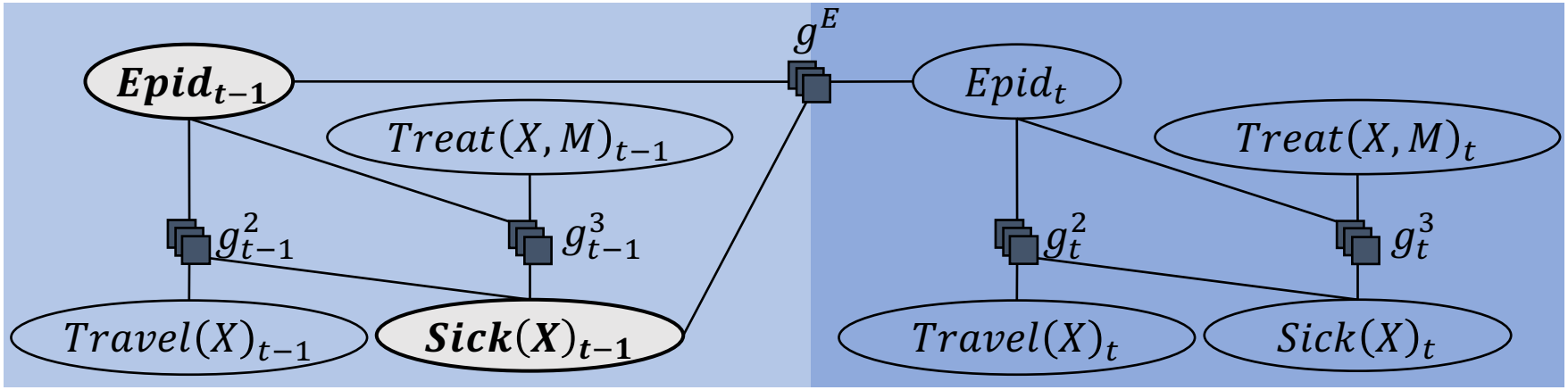
Time slices

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

PDM: Interfaces

- PRVs with index $t-1$ in transition parfactors
  - Called *forward* interface (separate $t-1$ from next slice t) but works in both directions
    - As each slice is a copy of the previous one, if the interface separates G_t from G_{t-1}, it also separates G_{t-1} from G_t
    - Backward interface: Find the PRVS that separate t from $t-1$
      - See Murphy (2002) for a discussion
- Formally, given a PDM $G = (G_0, G_{\rightarrow})$

  $\boldsymbol{I}_{t-1} = \{A_{t-1} | \exists g \in G_{\rightarrow} : (A_{t-1} \in rv(g) \wedge \exists A_t \in rv(g))\}$
  - Could also work with $G_{\rightarrow}^{1.5}$

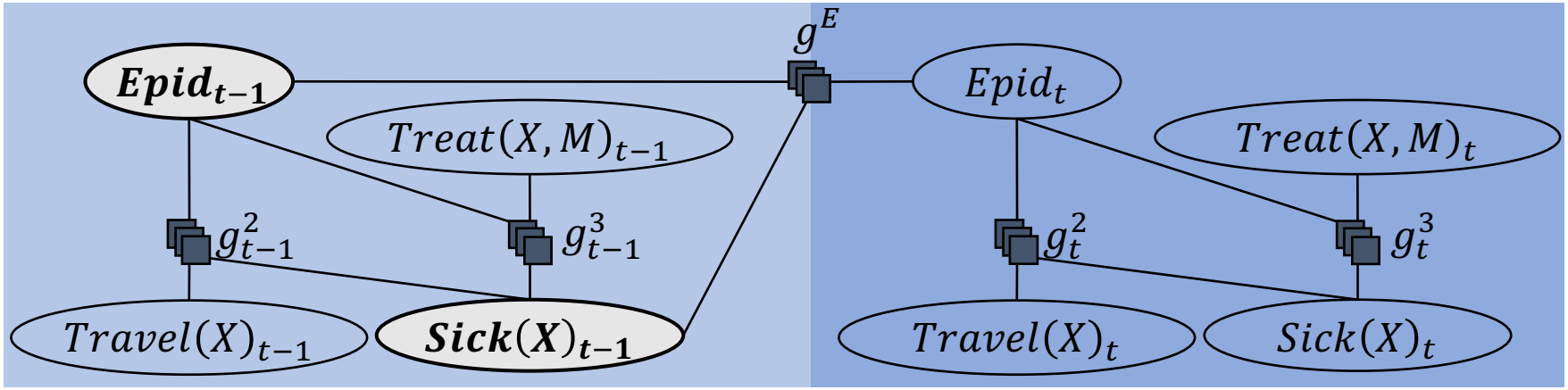


Time slices

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

PDM: Interfaces – Moving forward

- After one is finished with inference for τ, basically ask a query for $\boldsymbol{I}_\tau$ on the current model G_τ (as in a message in LJT) and include that information when instantiating a model $G_\rightarrow$ for $\tau + 1$
- Use FO jtrees and message passing
  - Make sure that $\boldsymbol{I}_{t-1}$ occurs in one parcluster
    - How? By adding a parfactor $g^I = \phi(\boldsymbol{I}_{t-1})$
      - By the FO jtree properties, there then is a parcluster containing $\boldsymbol{I}_{t-1}$
      - g^I can be removed after FO jtree construction
        - If not removed → ϕ has to be equally distributed

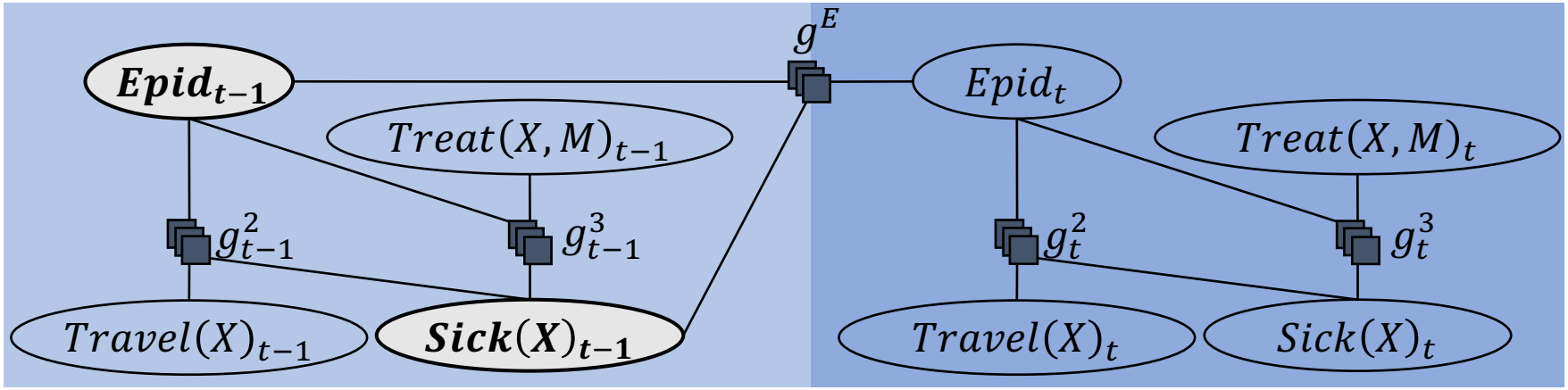


Time slices

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

PDM: Incluster & Outcluster

- Parcluster that contains $\boldsymbol{I}_{t-1}$ → incluster
  - From the perspective of t: separates past from present
  - Receives *in*coming information from outcluster of previous step
- Parcluster that contains $\boldsymbol{I}_t$ → outcluster
  - From the perspective of t: separates present from future
  - Sends information *out* to incluster of next step
- Present will become past for the next step so $\boldsymbol{I}_{t-1}$ and $\boldsymbol{I}_t$ needed



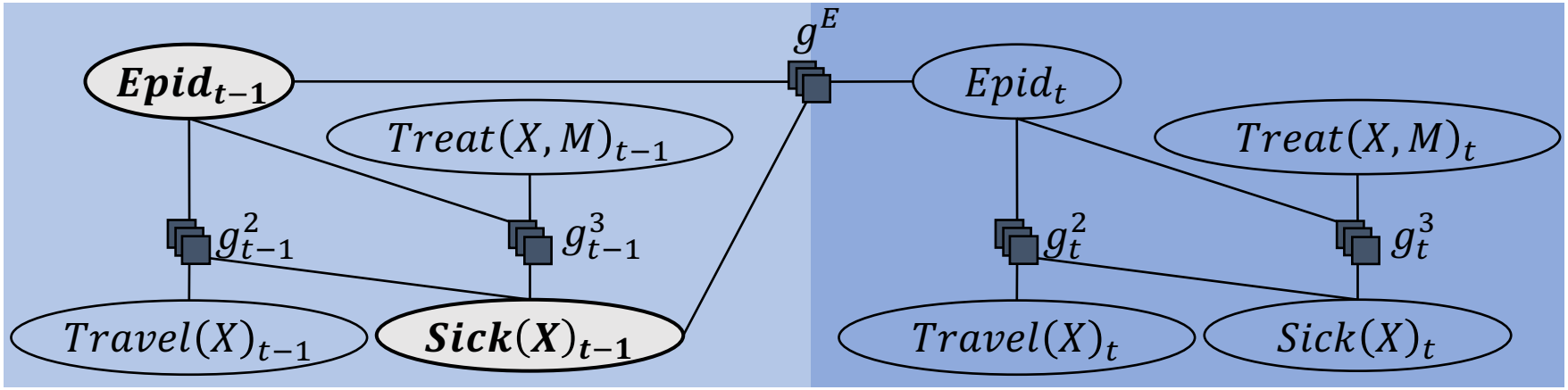
Time slices

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Dynamic FO Jtrees

- Given an interface $\boldsymbol{I}_{t-1}$ for a PDM $G = (G_0, G_{\rightarrow})$
- Build two FO jtrees $(J_0, J_{\rightarrow})$ for G
  - J_0 for $G_0 \cup \{\phi(\boldsymbol{I}_0)\}$
    - $\boldsymbol{I}_0$ in one parcluster (outcluster)
  - $J_{\rightarrow}$ for $G_{\rightarrow}^{1.5} \cup \{\phi(\boldsymbol{I}_{t-1}), \phi(\boldsymbol{I}_t)\}$
    - $\boldsymbol{I}_{t-1}$ in one parcluster (incluster)
    - $\boldsymbol{I}_t$ in one parcluster (outcluster)
  - Standard FO jtree construction works

An FO jtree $J = (V, E)$ consists of a set of parclusters V as nodes and a set of undirected edges E

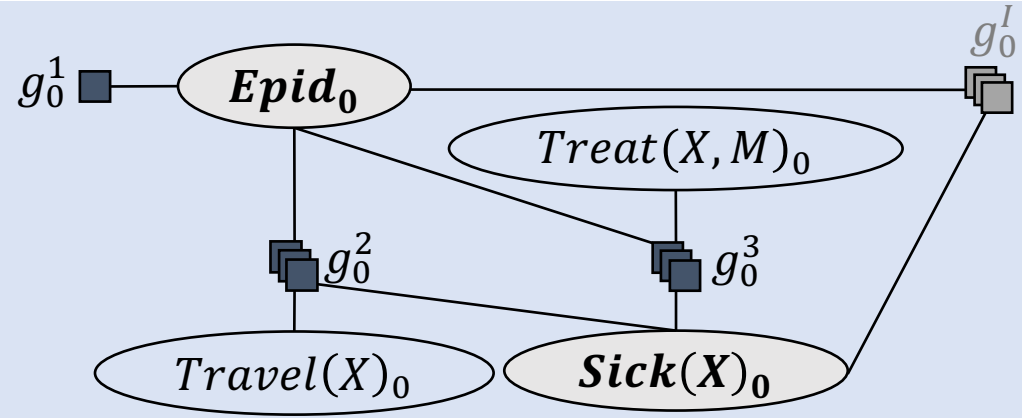


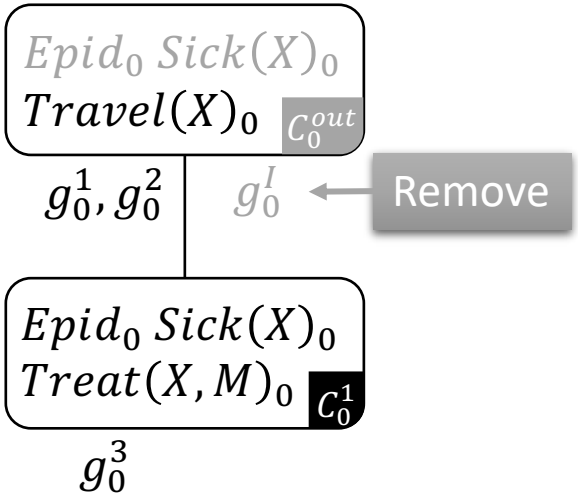
Time slices

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Dynamic FO Jtrees: Example

- J_0 for $G_0 \cup \{g_0^I\}$

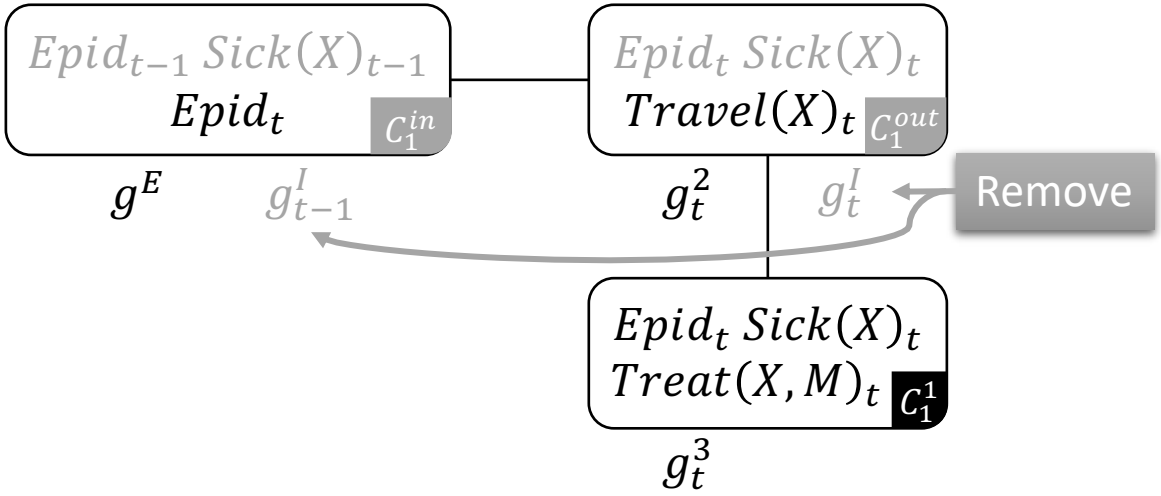


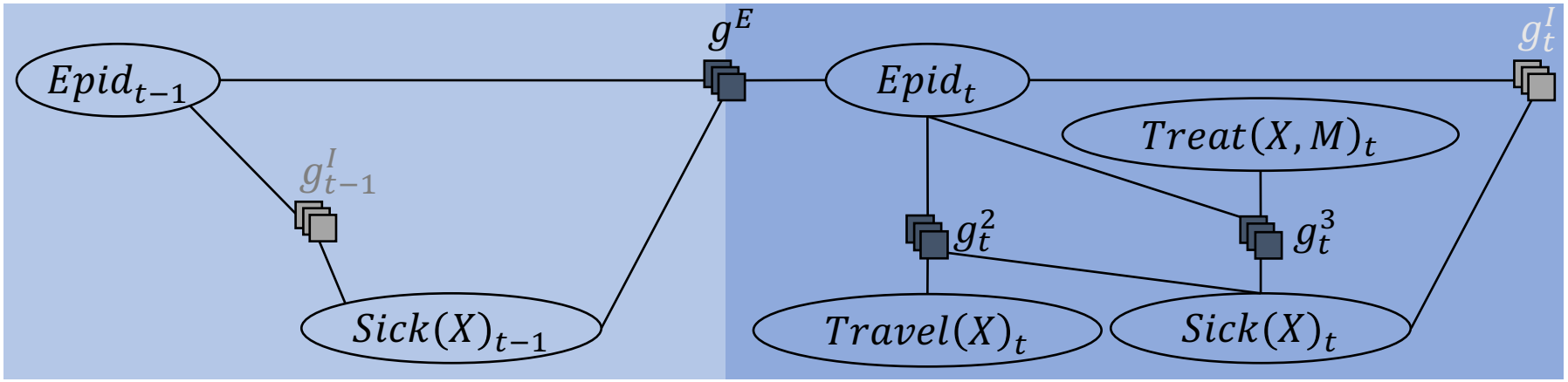


- Here, both parclusters contain the interface variables
  - Both could be outcluster
  - Outcluster determined by which parcluster contains g^I

Dynamic FO Jtrees: Example

- $J_{\rightarrow}$ for $G_{\rightarrow}^{1.5} \cup \{g_{t-1}^I, g_t^I\}$





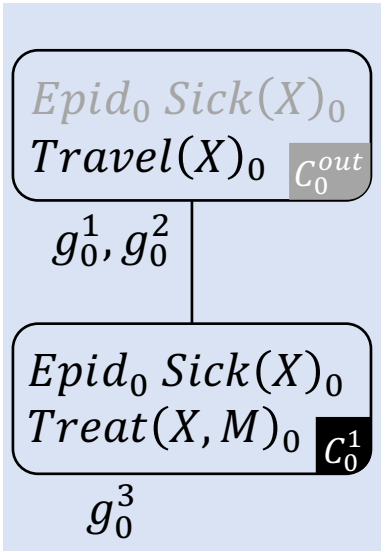
Dynamic FO Jtrees: Unrolling

- Given two FO jtrees $(J_0, J_\rightarrow)$ for G and a number T, unroll the FO jtree by
  - For $\tau = 0$, taking $J_0 = (V_0, E_0)$
  - For $\tau < T$, taking $J_\rightarrow = (V_\rightarrow, E_\rightarrow)$ instantiated for τ
  - For all $\tau - 1, \tau$, adding an edge between outcluster of $\tau - 1$ and incluster of τ
- Formally, $J_{0:T} = (V, E)$ where

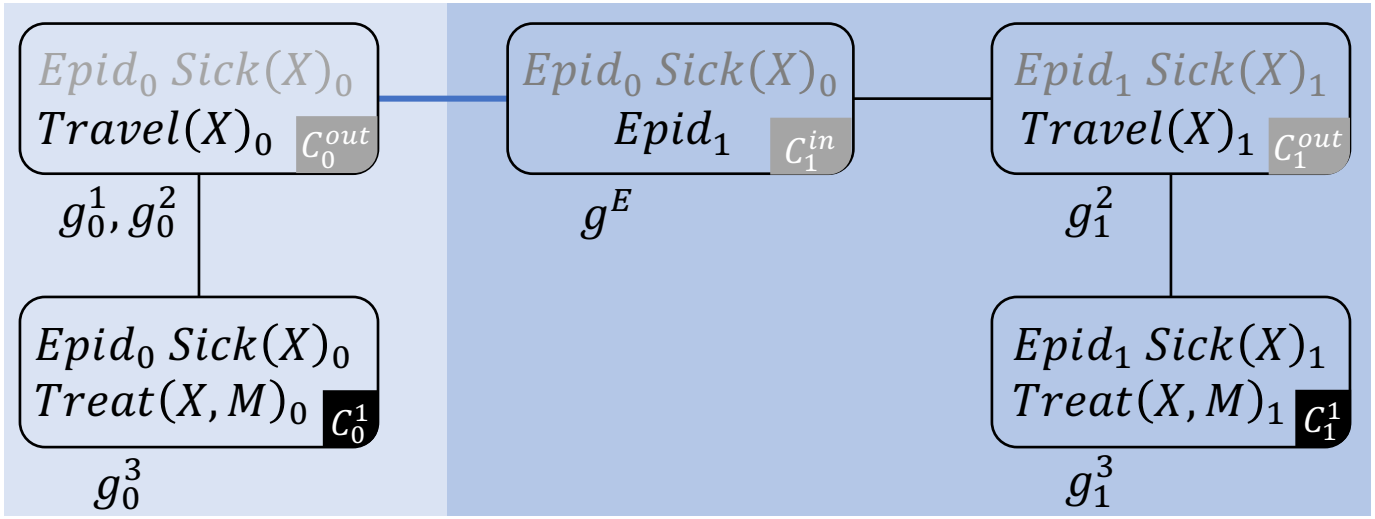
$$V = V_0 \cup \bigcup_{\tau=1}^{T} V_\tau$$

$$E = E_0 \cup \bigcup_{\tau=1}^{T} E_\tau \cup \bigcup_{\tau=1}^{T} \left\{ \{out, in\} \middle| C_{\tau-1}^{out} \wedge C_\tau^{in} \right\}$$

- $T = 2$
  - $\tau = 0$

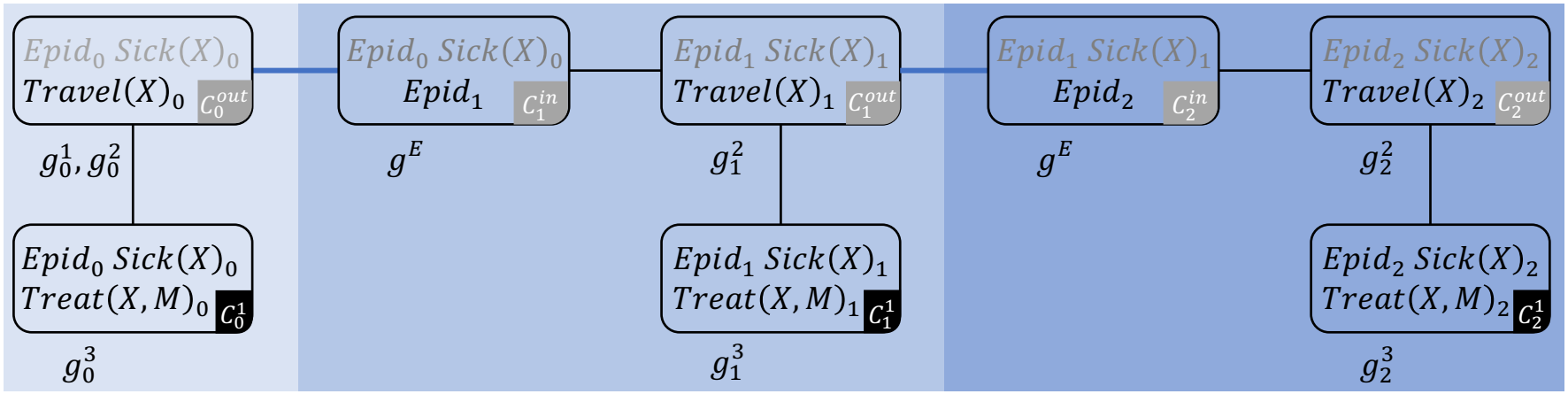

$Epid_0\ Sick(X)_0$
$Travel(X)_0$
C_0^{out}
g_0^1, g_0^2
$Epid_0\ Sick(X)_0$
$Treat(X,M)_0$
C_0^1
g_0^3

  - $\tau = 1$


$Epid_0\ Sick(X)_0$
$Travel(X)_0$
C_0^{out}
g_0^1, g_0^2
$Epid_0\ Sick(X)_0$
$Treat(X,M)_0$
C_0^1
g_0^3
$Epid_0\ Sick(X)_0$
$Epid_1$
C_1^{in}
g^E
$Epid_1\ Sick(X)_1$
$Travel(X)_1$
C_1^{out}
g_1^2
$Epid_1\ Sick(X)_1$
$Treat(X,M)_1$
C_1^1
g_1^3

Dynamic FO Jtrees: Unrolling – $T = 2$

- $\tau = T = 2$



- Unrolling unnecessary!
  - Because of the interface, current FO jtree J_τ enough to answer queries about current step τ
    - Inference within one step → LJT
    - Moving forward → send message from C_τ^{out} to $C_{\tau+1}^{in}$

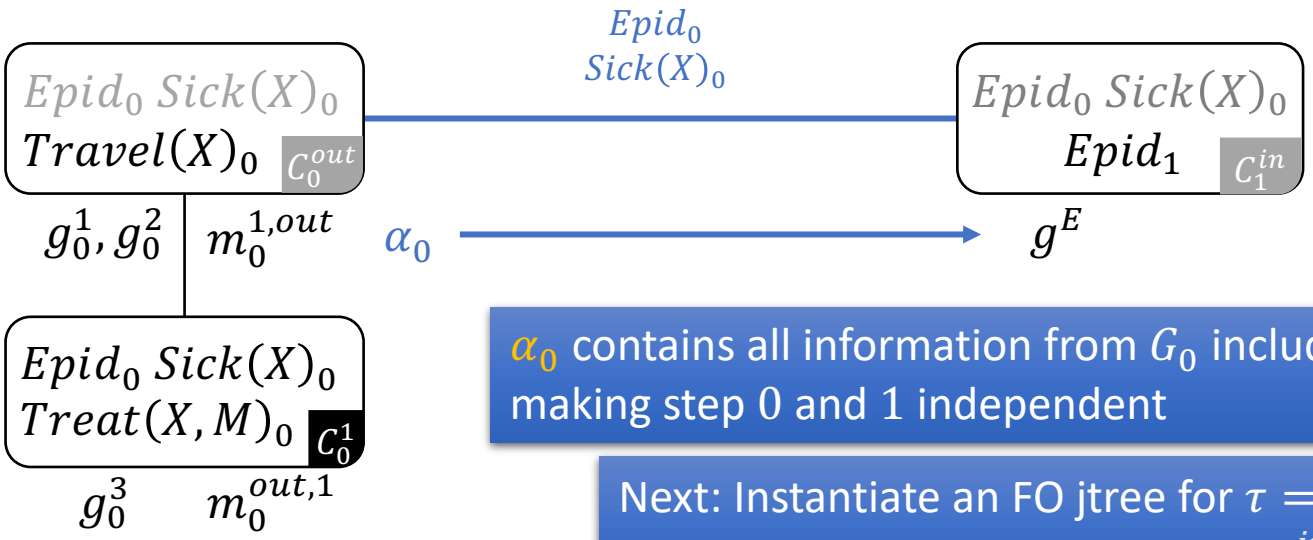
Moving Forward

- Inference within a step τ will follow LJT
  - One can ask a set of queries for τ efficiently
  - Ensures that all information is available at $\boldsymbol{C}_{\tau}^{out}$ to calculate the forward message that makes $\tau + 1$ independent from τ
- Moving forward:
  - Calculate a forward message α_{τ} over the separator between $\boldsymbol{C}_{\tau}^{out}$ and $\boldsymbol{C}_{\tau+1}^{in}$ using local model G_{τ}^{out} and messages $m_{\tau}^{j,out}$
  - Instantiate $J_{\rightarrow}$ for $\tau + 1$
  - Add α_{τ} to local model of $\boldsymbol{C}_{\tau+1}^{in}$
  - Drop J_{τ}
  - Perform inference in $J_{\tau+1}$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Moving Forward: Example

- $\tau = 0$
  - Current FO jtree
  - Intra-slice messages: $m_0^{1,out}$, $m_0^{out,1}$
  - Answer queries for $P(R_0|\boldsymbol{E}_0)$
- Inter-slice message: α_0
  - Eliminate non-separator PRV $Travel(X)_0$ from local model $G_0^{out} = \{g_0^1, g_0^2\}$ and $m_0^{1,out}$
  - Send result as message α_0 to C_1^{in}

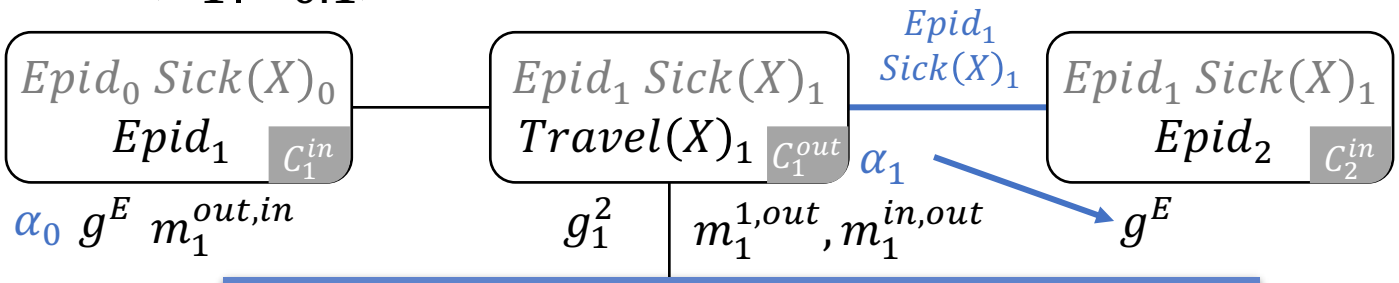


α_0 contains all information from G_0 including $\boldsymbol{E}_0$, making step 0 and 1 independent

Next: Instantiate an FO jtree for $\tau = 1$ and add α_0 to the local model of C_1^{in}

Moving Forward: Example

- $\tau = 1$
  - Current FO jtree
  - Intra-slice messages: $m_0^{out,in}$, $m_1^{in,out}$, $m_0^{1,out}$, $m_0^{out,1}$
  - Answer queries for $P(R_1|\boldsymbol{E}_{0:1})$
- Inter-slice message: α_1
  - Eliminate non-separator PRV $Travel(X)_1$ from local model $G_1^{out} = \{g_1^2\}$ and $m_1^{1,out}, m_1^{in,out}$
  - Send α_1 to C_2^{in}

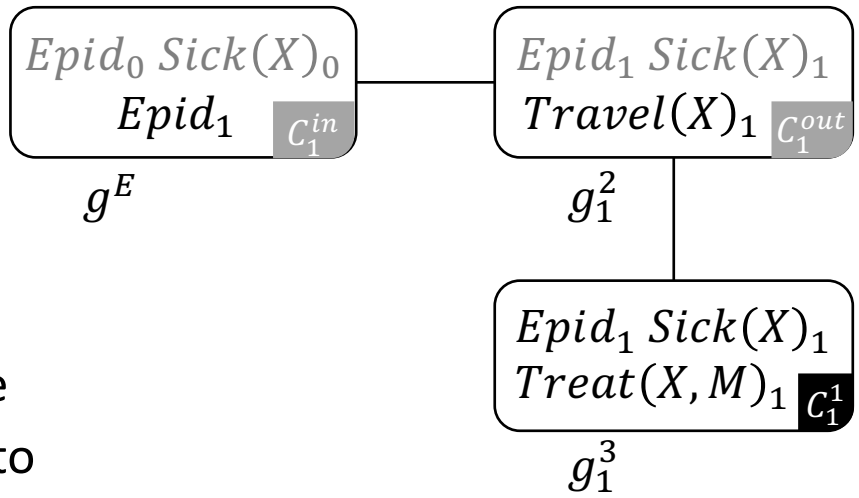


During message passing, the information in α_0 is passed around as well and therefore also included when calculating α_1.
α_1 now includes information of $G_{0:1}$ including $\boldsymbol{E}_{0:1}$, making step 1 and 2 independent.

Next: Instantiate an FO jtree for $\tau = 2$ and add α_1 to the local model of C_2^{in}

Filtering

- Queries answered in this fashion: filtering queries
  - $P(R_\tau | \boldsymbol{E}_{0:\tau})$
  - Queries for the current step given all previous evidence
  - Upsides
    - Only one current FO jtree
    - One additional message to move forward



- What about prediction and hindsight?
  - $P(R_\pi | \boldsymbol{E}_{0:\tau})$, $\pi \neq \tau$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

Prediction

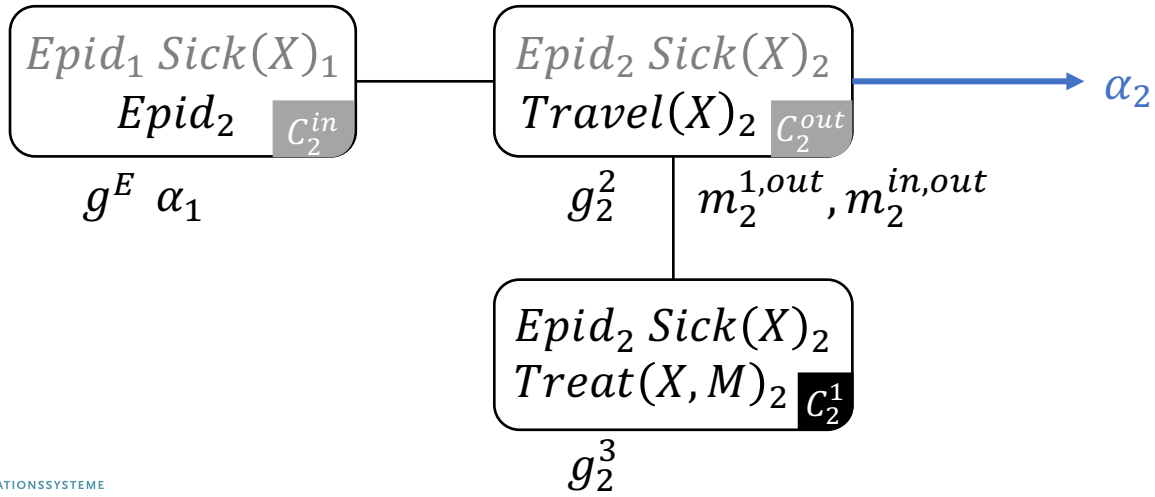
- Prediction queries for future PRV instances
  - $P(R_\pi|\boldsymbol{E}_{0:\tau}), \pi > \tau$
  - Flashing into the future, i.e., moving forward to π without seeing evidence in between, i.e.,
    - Filtering query $P(R_\pi|\boldsymbol{E}_{0:\pi})$ where the evidence sets between $\tau+1$ and π in $\boldsymbol{E}_{0:\pi}$ are empty: $\boldsymbol{E}_{\tau+1:\pi} = \{\emptyset_{\tau'}\}_{\tau'=\tau+1}^{\pi}$
- Prediction query answering
  - Given τ current step
  - Move forward until $\tau = \pi$
    - When moving forward, only an inbound message pass with the outcluster as centre is necessary
  - Answer query with query term R_π
    - If only one query, locate parcluster $\boldsymbol{C}_\pi^i$ containing R_π and trigger one inbound message pass with $\boldsymbol{C}_\pi^i$ as centre
    - Otherwise, perform complete message passing

Saves half of the messages

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *ICCS-18 Proceedings of the International Conference on Conceptual Structures*, 2018.

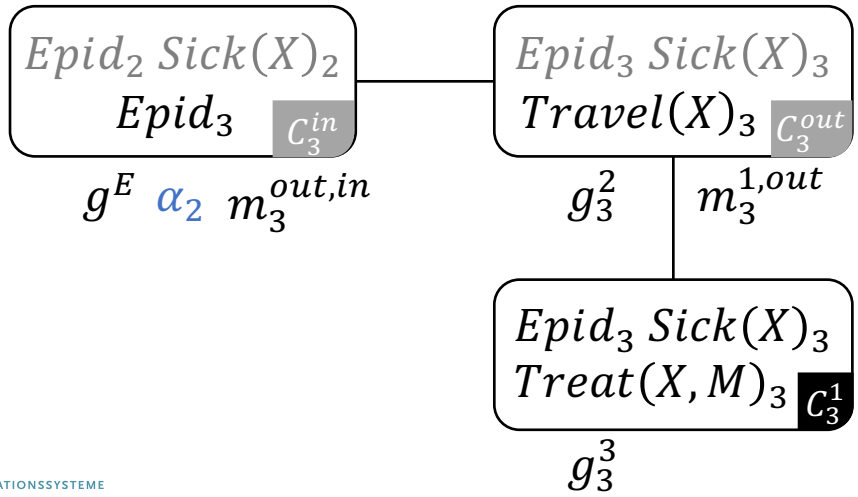
Prediction: Example

- $P(Epid_3|\boldsymbol{E}_{0:1}), 3 > 1$
  - No evidence for steps 2 and 3: $\boldsymbol{E}_{2:3} = \{\emptyset_{\tau'}\}_{\tau'=2}^{3}$
  - Given $\tau = 1$ current step, move forward until $\tau = 3$, then answer query with query term $Epid_3$
  - $\tau = 2$, current FO jtree with α_1 in local model of C_2^{in}
    - Send intra-slice messages towards C_2^{out} (collect all information at outcluster); also, no evidence entering as $\emptyset_2$
    - Calculate inter-slice message α_2



Prediction: Example

- $P(Epid_3|\boldsymbol{E}_{0:1}), 3 > 1$
  - $\tau = 3$, current FO jtree with α_2 in local model of C_3^{in}
  - If answering only $Epid_3$, collect messages at one of the parclusters, e.g., C_3^{in} and answer query
    - If so, no need to also calculate $m_3^{in,out}, m_3^{out,1}$



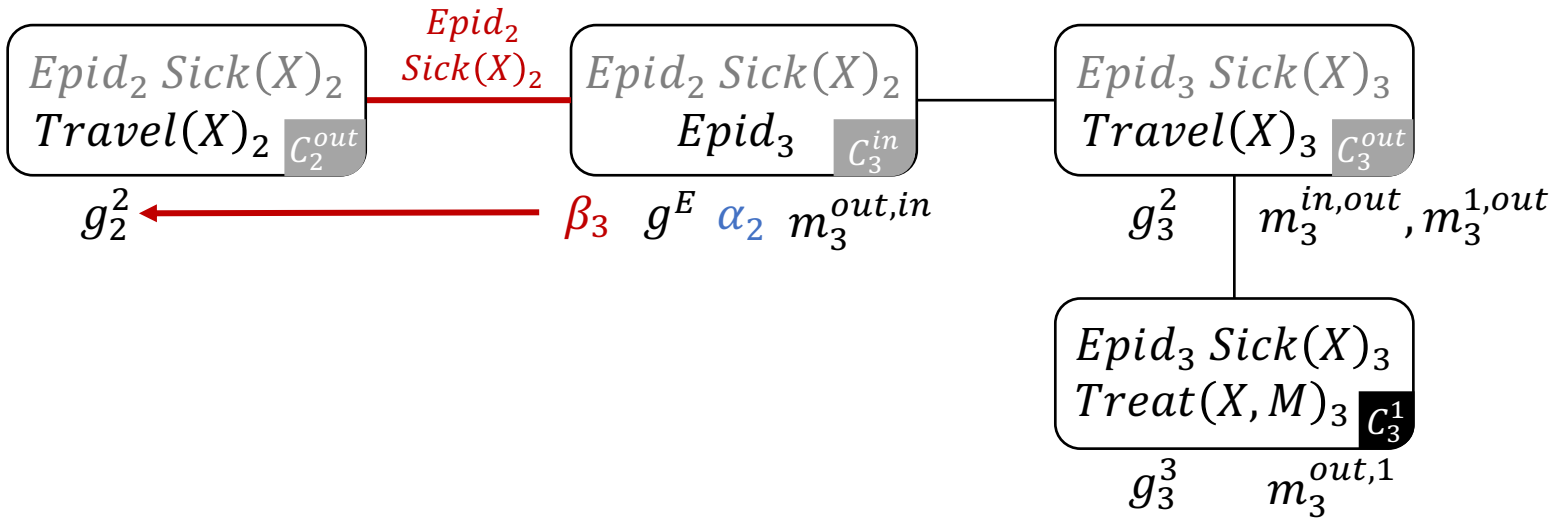
Hindsight: Moving Back Again

- Hindsight queries for past PRV instances
  - $P(R_\pi | \boldsymbol{E}_{0:\tau}), \pi < \tau$
  - Looking back given what we know now in terms of evidence, i.e.,
    - From the current step τ, we have to go back and send the information accumulated between π and τ to π and then answer a filtering query in step π again
- Hindsight query answering
  - Given τ current step
  - Move backward until $\tau = \pi$
    - Calculate *backward messages* β_τ starting from τ until π
      - When moving backward, only an inbound message pass with the incluster as centre is necessary
  - Answer query with query term R_π

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

Hindsight: Backward Message β_τ

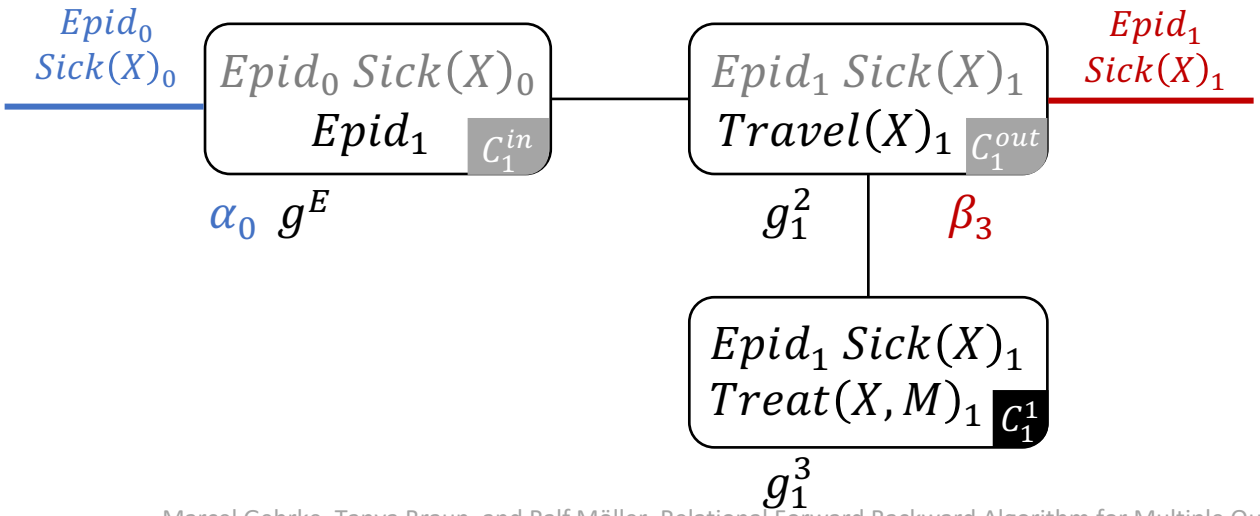
- Same as forward but sent from $\boldsymbol{C}_\tau^{in}$ to $\boldsymbol{C}_{\tau-1}^{out}$
  - Forward message α_τ contains all information on $G_{0:\tau}$ including $\boldsymbol{E}_{0:\tau}$, making $\tau+1$ independent from τ
  - Backward message β_τ contains all information on $G_{\tau:T}$ including $\boldsymbol{E}_{\tau:T}$, making $\tau-1$ independent from τ
    - Exclude forward message $\alpha_{\tau-1}$ for calculation, as that information is already present at slide τ
- E.g., calculate β_3 by eliminating $Epid_3$ from $g^E, m_3^{out,in}$ (without α_2)



Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

Hindsight: Independences

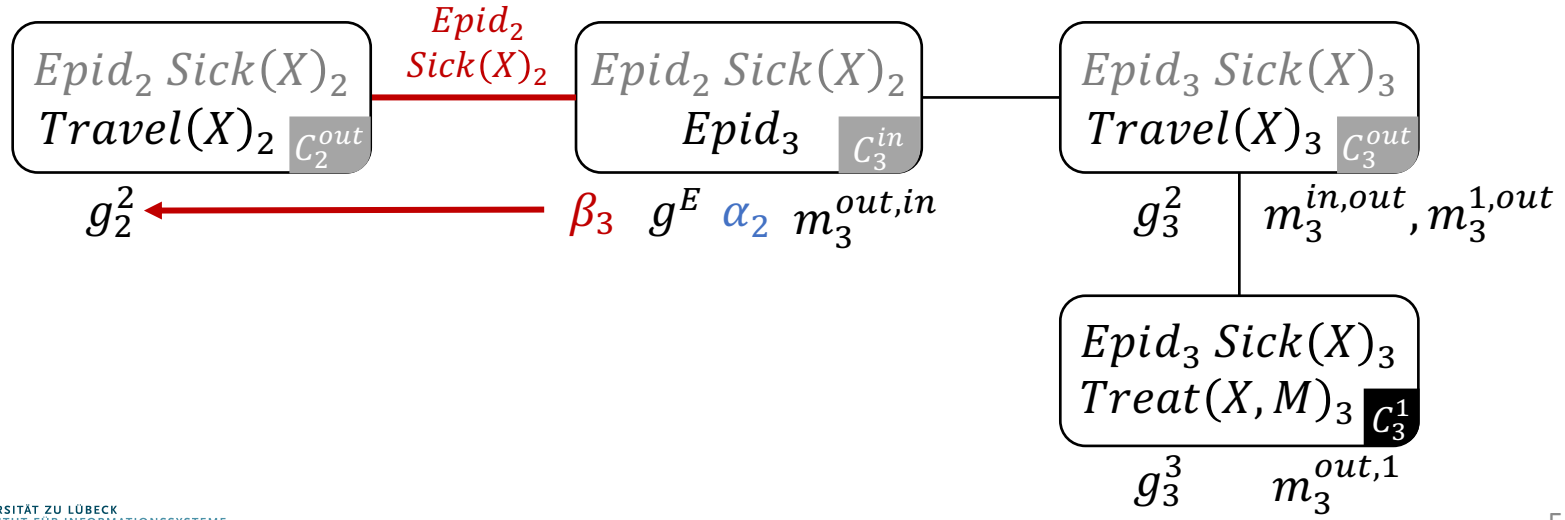
- Given $P(R_\pi | \boldsymbol{E}_{0:\tau})$, $\pi < \tau$, at goal slice π,
  - Forward message $\alpha_{\pi-1}$ contains all information on $G_{0:\pi-1}$ including $\boldsymbol{E}_{0:\pi-1}$, making π independent of $\pi - 1$
    - Requires storing forward messages α_τ; otherwise, one would have to re-do moving forward from 0 to π
  - Backward message $\beta_{\pi+1}$ contains all information on $G_{\pi+1:\tau}$ including $\boldsymbol{E}_{\pi+1:\tau}$, making π independent of $\pi + 1$



Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

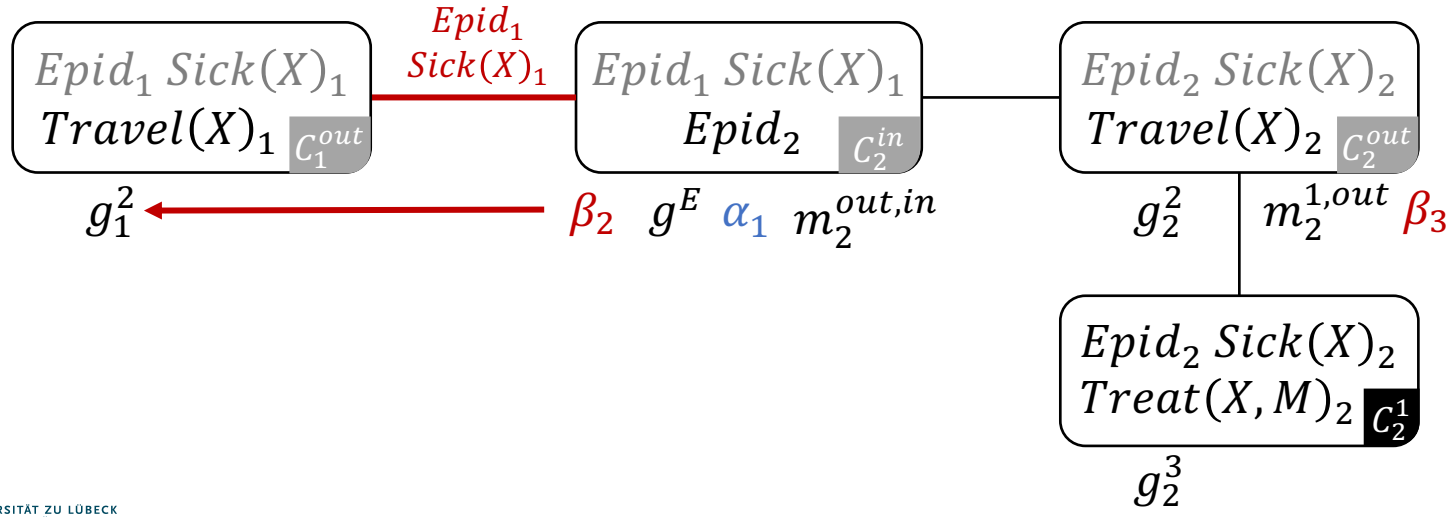
Hindsight: Example

- $P(Epid_1|\boldsymbol{E}_{0:3}), 1 < 3$
  - Given $\tau = 3$ current step, move backward until $\tau = 1$, then answer query with query term $Epid_1$
  - $\tau = 3$, current FO jtree with α_2 in local model of C_3^{in}
    - Calculate backward message β_3
    - Instantiate an FO jtree for $\tau = 2$, add β_3 to local model of C_2^{out}



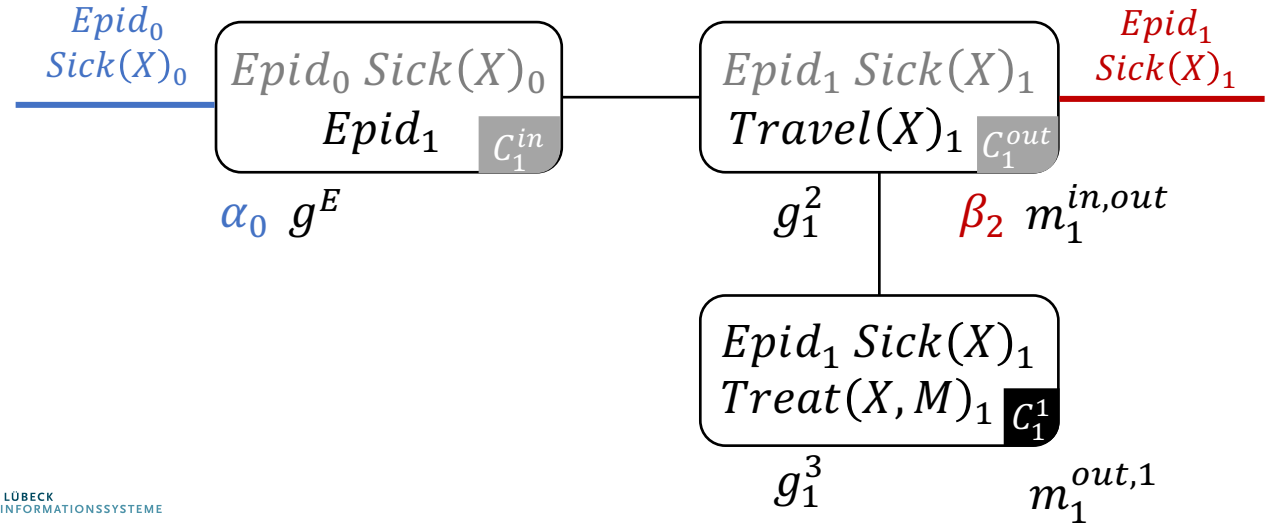
Hindsight: Example

- $P(Epid_1|\boldsymbol{E}_{0:3}), 1 < 3$
  - $\tau = 2$, current FO jtree with β_3 in local model of C_2^{out}
    - Enter evidence $\boldsymbol{E}_2$, send messages inbound to C_2^{in}
    - Calculate backward message β_2, ignoring α_1
    - Instantiate an FO jtree for $\tau = 1$, add β_2 to local model of C_1^{out}



Hindsight: Example

- $P(Epid_1|\boldsymbol{E}_{0:3}), 1 < 3$
  - $\tau = 1$, current FO jtree with β_2 in local model of C_1^{out}
    - Enter evidence $\boldsymbol{E}_1$
    - If answering only $Epid_1$, collect messages at one of the parclusters, e.g., C_1^1 and answer query
      - If so, no need to also calculate $m_1^{out,in}, m_1^{1,out}$



General Query Answering

- Relational forward-backward algorithm for all types of queries, Filtering, prediction, hindsight
- Set of queries $\boldsymbol{Q}_{0:T_q} = \left\{ \left\{ Q_{\pi_i}^i \right\}_{i=1}^{m_t} \right\}_{t=0}^{T_q}$
  - Basically a stream of queries
  - For efficiency, go through $\boldsymbol{Q}_{0:T_q}$ by t and through each $\left\{ Q_{\pi_i}^i \right\}_{i=1}^{m_t}$ by type of query and π_i
    1. Filtering queries
    2. Prediction queries, ordered by increasing π_i
    3. Hindsight queries, ordered by decreasing π_i
    - Order of 2. and 3. could be exchanged

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

General Query Answering

- Given τ as the current step with query terms $\{Q_{\pi_i}^i\}_{i=1}^{m_\tau}$ and a completed message pass for τ
  - For all $Q_{\pi_i}^i$ where $\tau = \pi_i$, answer $Q_{\pi_i}^i$ in J_τ
  - For all $Q_{\pi_i}^i$ where $\tau < \pi_i$, move forward without evidence until $\max_i \pi_i$
    - Instantiate FO jtrees and calculate messages accordingly
    - Whenever $\tau = \pi_i$ while moving forward, answer $Q_{\pi_i}^i$
  - For all $Q_{\pi_i}^i$ where $\tau > \pi_i$, move backward until $\min_i \pi_i$
    - Instantiate FO jtrees, enter evidence, and calculate messages accordingly
    - Whenever $\tau = \pi_i$ while moving backward, answer $Q_{\pi_i}^i$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

Lifted Dynamic Jtree Algorithm (LDJT)

procedure LDJT$((G_0, G_{\rightarrow}), \boldsymbol{Q}_{0:T_q}, \boldsymbol{E}_{0:T_e})$

Build 1.5-slice model $G_{\rightarrow}^{1.5}$

Construct FO jtree $(J_0, J_{\rightarrow})$ for G_0 and $G_{\rightarrow}^{1.5}$

for τ **in** $0 \ldots T_q$ **do**

Instantiate J_τ

Add $\alpha_{\tau-1}$ to incluster of J_τ ▹ if $\tau > 0$

Enter evidence $\boldsymbol{E}_\tau$ into J_τ

Pass messages in J_τ

Answer queries $\boldsymbol{Q}_\tau$

Calculate α_τ

LDJT

- In online inference, two streams, one for incoming evidence, one for incoming queries instead of inputs $\boldsymbol{Q}_{0:T_q}, \boldsymbol{E}_{0:T_e}$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

Approaches to Backward Passes

- Backward passes as presented so far require recalculating some of the messages
  - Each step backwards without queries for a step
    - Calculate an inbound message pass with $n-1$ messages, n the number of parclusters in J_τ
      - Collect model information and evidence for τ as well as future information in $\beta_{\tau+1}$
  - Each step backwards with queries for a step
    - Calculate a complete message pass with $2(n-1)$ messages, n the number of parclusters in J_τ
  - BUT: only one FO jtree in memory at a time
    - For hindsight queries, forward messages need to be stored!
- What if LDJT does not drop FO jtrees once τ moves forward?
  - Keep instantiations vs. instantiating them on demand

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

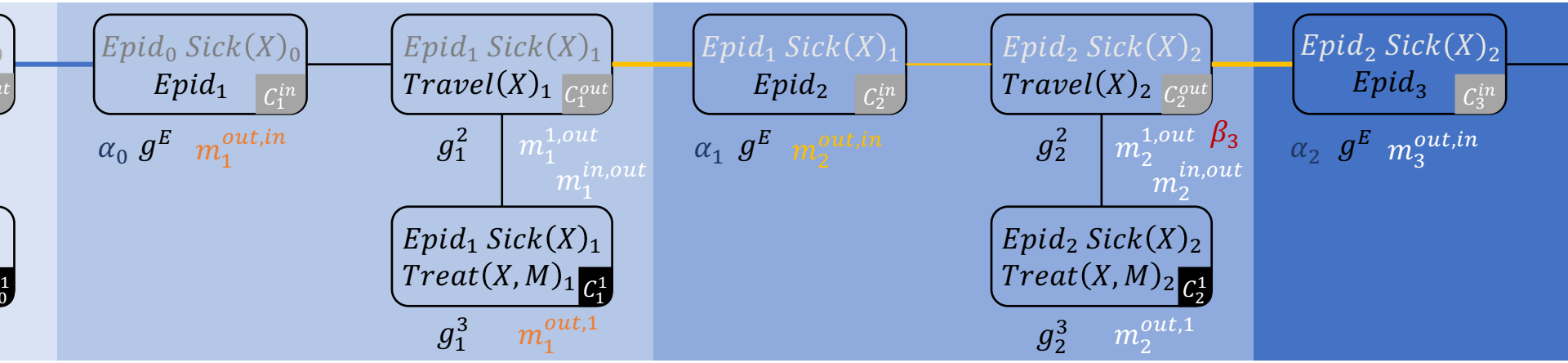
Keep Instantiations

- What does LDJT pay in memory?
  - Additionally store local models, intra-slice messages for each slice
- How much runtime can LDJT save?
  - C.f., adaptive message passing: Only update those messages affected by new incoming information in $\beta_{\tau+1}$
  - Each step backwards without queries for a step
    - Calculate messages on path between outcluster and incluster of $J_\tau \rightarrow$ up to $n-1$ messages, n the number of parclusters in J_τ
  - Each step backwards with queries for a step
    - Calculate an outbound message pass with $n-1$ messages, n the number of parclusters in J_τ

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.

Example: $P(Epid_1|\boldsymbol{E}_{0:3})$

- In $\tau = 2$ (no queries)
  - Keep instantiations: $m_2^{1,out}$ (on path)
  - Ondemand: $m_2^{1,out}, m_2^{out,in}$ (inbound to C_2^{in})
- In $\tau = 1$ (with multiple queries)
  - Keep instantiations: $m_1^{out,1}, m_1^{out,in}$ (outbound)
  - Ondemand: $m_1^{out,1}, m_1^{out,in}, m_1^{1,out}, m_1^{in,out}$



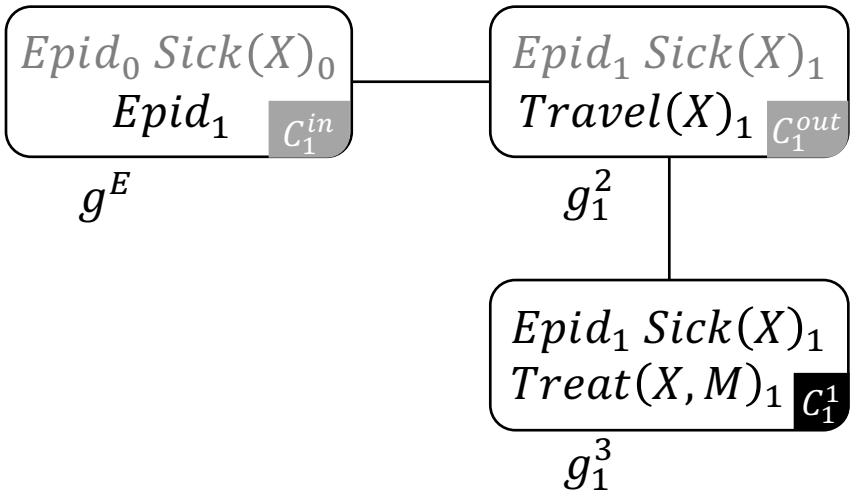
Basically the unrolled FO jtree in memory

Approaches to Backward Passes

- Keeping all instantiations in memory not sustainable
  - Aim was to have a memory-efficient algorithm that does not need the complete model from the first step onwards
- Could trade off runtime vs. memory requirements
  - Keep the last k instantiations
    - Analyse data/queries for typical lags
  - Instantiate on demand for queries with large lags

	Keep instantiations	**Instantiate on demand**
Steps without queries	$\leq n-1$	$n-1$
Steps with queries	$n-1$	$2(n-1)$
Additional memory	Local models + messages	Only α_t messages

Marcel Gehrke, Tanya Braun, and Ralf Möller. Relational Forward Backward Algorithm for Multiple Queries. In *FLAIRS-32 Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*, 2019.



Beyond Standard LDJT

Algorithm-induced groundings with LDJT

Conjunctive queries in PDMs, MPE/MAP queries in PDMs,

Complexity analysis, Implementation/runtimes

Algorithm-induced Groundings

- Like LJT, LDJT may induce groundings during message passing
  - Intra-slice: use LJT-fusion to avoid
    - Merge parclusters if a separator PRV induces groundings

Conditions on Groundings

- For a lifted calculation of message m_{ij}, it necessarily has to hold that
  - for each PRV $A \in (\boldsymbol{C}_i \setminus \boldsymbol{S}_{ij})$, i.e., A has to be eliminated:
    - for each separator PRV $S \in \boldsymbol{S}_{ij}$: $lv(S) \subseteq lv(A)$ (Cond. 1)
- If Cond. 1 does not hold, i.e., $lv(S) \not\subseteq lv(A)$, one may induce Cond. 1 by count conversion
  - If $lv(S) \setminus lv(A)$ are countable in G_{ij} (Cond. 2)

Fusion

- Extra step at end of construction called fusion
  - Test each possible message m_{ij} for each PRV A to eliminate and each separator PRV S based on the three conditions
    - If Cond. 1 holds: no groundings for A and S; continue
    - Otherwise:
      - If Cond. 2 holds: check Cond. 3
        - If Cond. 3 holds: no groundings for A and S; continue
        - Otherwise: groundings; mark m_{ij}; continue with next message
      - Otherwise: groundings; mark m_{ij}; continue with next message
  - For each message m_{ij} marked:
    - Merge parclusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ (as in minimisation)

Fusion

Conditions on Groundings

- Problem with induced Cond. 1 using count conversions on the logvars in $lv(S) \setminus lv(A)$:
  - Logvars that were previously not counted are now counted
  - All receiving parclusters need to be able to handle the counted versions, which needs to be checked
    - If a newly counted logvar arrives at a parcluster $\boldsymbol{C}_k$, it has to be countable in G_k as well (Cond. 3)
      - For further calculations, since they refer to the same set of randvars, they have to occur in the same form, i.e., at one point the logvar has to be counted in $\boldsymbol{C}_k$ as well

Algorithm-induced Groundings

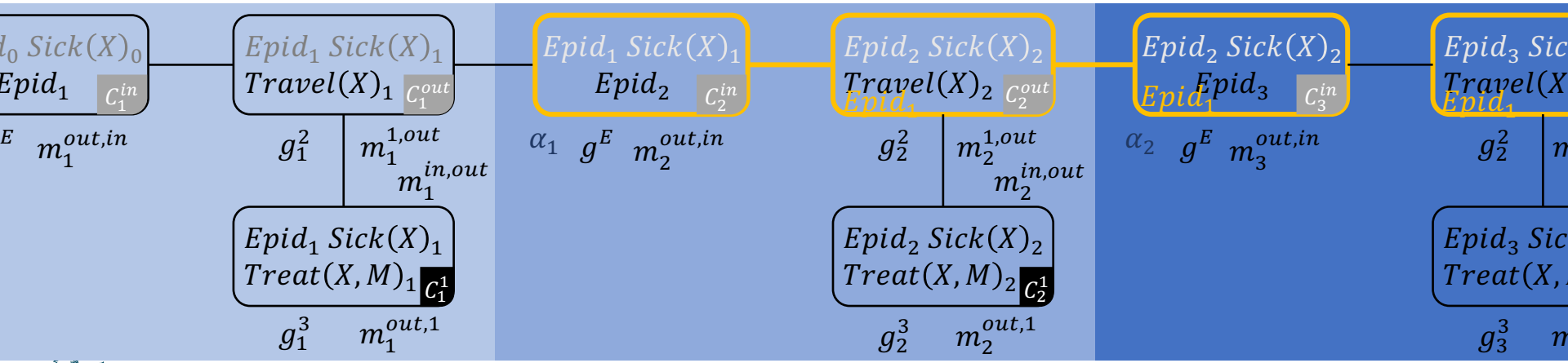
- Inter-slice?
  - Check if calculation of α_t leads to groundings using the same conditions as LJT-fusion
    - If so, extend incluster with PRV that causes groundings
      - Do not merge but keep separate to keep up sequential/temporal separation between t and $t+1$
  - But: still may lead to outcluster of t or incluster of $t+1$ being a subset of the other (separation removed) that cannot be avoided if wanting to avoid groundings

→Trade off between lifting and handling temporal aspects due to restrictions on elimination orders

Marcel Gehrke, Tanya Braun, and Ralf Möller. Preventing Unnecessary Groundings in the Lifted Dynamic Junction Tree Algorithm. In: *Proceedings of the AI 2018: Advances in Artificial Intelligence*, 2018.
Marcel Gehrke, Tanya Braun, Ralf Möller: Towards Preventing Unnecessary Groundings in the Lifted Dynamic Junction Tree Algorithm. In: *Proceedings of KI 2018: Advances in Artificial Intelligence*, 2018.

Conjunctive Queries

- May contain query terms that cover multiple steps
  - E.g., $P(Epid_1, Travel(X)_3)$
- Conjunctive query answering
  - Find subgraph covering all query terms (like LJT)
    - Unroll FO jtrees for steps occurring in query
  - Take parcluster $\boldsymbol{C}$ with largest overlap with query terms
  - Add all query terms to $\boldsymbol{C}$
  - Ensure running intersection property by extending parclusters
  - Perform an inbound message pass with $\boldsymbol{C}$ at centre and answer query at $\boldsymbol{C}$



Marcel Gehrke, Tanya Braun, and Ralf Möller. Answering Multiple Conjunctive Queries with the Lifted Dynamic Junction Tree Algorithm. In: *Proceedings of the AI 2018: Advances in Artificial Intelligence*, 2018.

Sequential MAP (and MPE)

- Most probable assignments to PRVs $\boldsymbol{U}_{|C'}$ given evidence $\boldsymbol{e}$

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

- Now, queries for a most probable sequence
- Procedure (like MAP-LJT)
  - Find a subgraph covering $\boldsymbol{U}_{|C'}$
  - Eliminate $\boldsymbol{T}_{|C''}$ using LVE operators in model of subgraph
  - Eliminate $\boldsymbol{U}_{|C'}$ using MPE-LVE operators

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Temporal Most Probable Explanation. In *ICCS-19 Proceedings of the International Conference on Conceptual Structures*, 2019.

Sequential MAP (and MPE)

- Most probable assignments to PRVs $\boldsymbol{U}_{|C'}$ given evidence $\boldsymbol{e}$

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

- If MAP query but over complete time steps
  - MPE for all PRVs in $G_{\tau_1:\tau_2}$ given evidence $\boldsymbol{E}_{0:\tau_2}$
  - Starting at $\tau = \tau_1$ with $\alpha_{\tau-1}$ calculated as before
    - Move forward until $\tau = \tau_2$ using MPE-LVE operators in calculations
      - Inbound message passes with outclusters as centre

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Temporal Most Probable Explanation. In *ICCS-19 Proceedings of the International Conference on Conceptual Structures*, 2019.

Complexity

- LJT complexity for message passing and query answering

Lifted calculations Largest parfactor size

$$O_{MP} = O\left(n_J \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right)$$

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

  - n_T, n_J number of nodes in FO dtree/jtree
  - n largest domain size
  - r largest range size, $r_\#$ largest range size of a PRV in a CRV
  - w_g largest ground width
  - $w_\#$ largest counting width

- LDJT: Moving forward in time
  - Uses message passing within each time step → O_{MP}
  - Uses LJT query answering for inter-slice message → O_{QA}

What are the worst-case and best-case scenarios in terms of queries?

Complexity

- Given a maximum number T occurring over all M queries ($M = \sum_{t=0}^{T} m_t, m = \frac{1}{T}\sum_{t=0}^{T} m_t$)
  - Moving forward: $T \cdot \left(O_{MP} + O_{QA}\right)$
  - *Best* worst case at a time step $\tau = \tau' \in \{0, \dots, T\}$
    - Filtering query for τ', each with O_{QA}
  - Overall,

$$
\begin{aligned}
& T \cdot \left(O_{MP} + O_{QA}\right) + M \cdot O_{QA} \\
&= \left(T \cdot n_J + T + M\right) \cdot O\left(\log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right) \\
&= O\left(\left(T \cdot n_J + M\right) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right) \\
&= O\left(\left(T \cdot n_J + T \cdot m\right) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right)
\end{aligned}
$$

Compare LJT complexity

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

Complexity

- *Worst* worst case at a time step $\tau = \tau' \in \{0, \dots, T\}$
  - Hindsight query for $\tau = 0$
    - Backward message pass to $\tau = 0$: $\tau' \cdot (O_{MP} + O_{QA})$
  - Prediction query for $\tau = T$
    - Forward message pass to $\tau = T$: $(T - \tau') \cdot (O_{MP} + O_{QA})$
  - Together, a hindsight and a prediction query yield a complexity of

$$\tau' \cdot (O_{MP} + O_{QA}) + (T - \tau') \cdot (O_{MP} + O_{QA})$$
$$= T \cdot (O_{MP} + O_{QA})$$

  - For m queries per step, we have

$$T \cdot (O_{MP} + O_{QA}) + m \cdot O_{QA}$$

- For T steps, we have

$$T \cdot T \cdot (O_{MP} + O_{QA}) + T \cdot m \cdot O_{QA}$$
$$= T^2 \cdot (O_{MP} + O_{QA}) + M \cdot O_{QA}$$

Complexity

- *Worst* worst case at a time step $\tau = \tau' \in \{0, \dots, T\}$
- Moving forward: $T \cdot (O_{MP} + O_{QA})$
- Query answering complexity

$$T^2 \cdot (O_{MP} + O_{QA}) + M \cdot O_{QA}$$

- Overall,

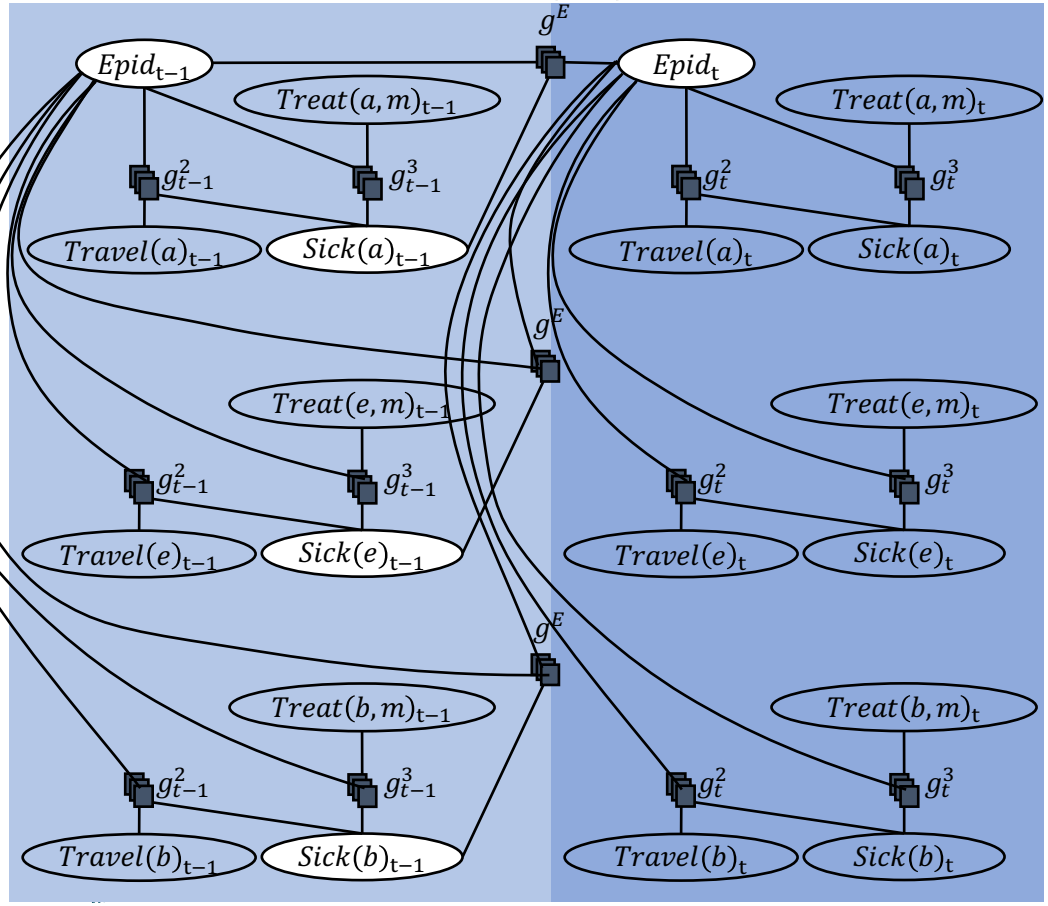
$$T \cdot (O_{MP} + O_{QA}) + T^2 \cdot (O_{MP} + O_{QA}) + M \cdot O_{QA}$$
$$= (T \cdot n_J + T + T^2 \cdot n_J + T^2 + M) \cdot O_{QA}$$
$$= O\left(\left((T^2 + T) \cdot n_J + M\right) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right)$$
$$= O\left(\left((T^2 + T) \cdot n_J + T \cdot m\right) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right)$$

Compare filtering complexity

$$O\left((T \cdot n_J + T \cdot m) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right)$$

Comparison to Ground Inference

- Grounding with
  - $\mathcal{D}(X) = \{a, e, b\}$
  - $\mathcal{D}(M) = \{m\}$



- Earning of LVE vs. VE
  - $n_{gr(T)} \gg n_T$ (with large domains)
  - $w \gg (w_g + w_\#)$ (with count conversions)
    - Without count conversions, $w = w_g$ and $w_\# = 0$
- In sequential case,
  - Earnings because of lifting the interface
    - Even without count conversions, $w \gg w_g$

Runtimes

- Algorithms for comparison
  - LDJT as presented here
  - DJT: propositional interface algorithm
  - LJT FOJT: unrolling the FO jtree
    - Performs similar to LDJT as they perform the same calculations only LJT FOJT has to keep unrolled model in memory
  - LJT Model: unrolling the model
- X-axis: maximum number of steps T
- Y-axis: runtime in seconds
- Figs. 1 + 2: Filtering queries
- Fig. 3: Hindsight queries
  - Prediction runtimes look similar
- Fig. 4: Preventing groundings

Implementation available at:
https://www.ifis.uni-luebeck.de/index.php?id=483

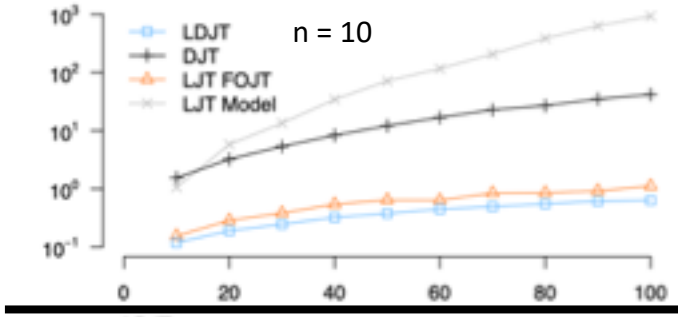


Fig. 1

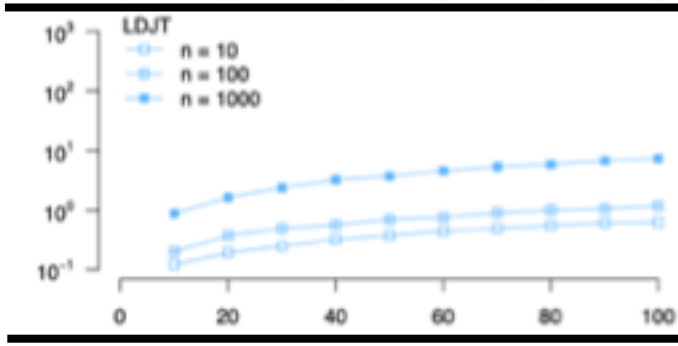


Fig. 2

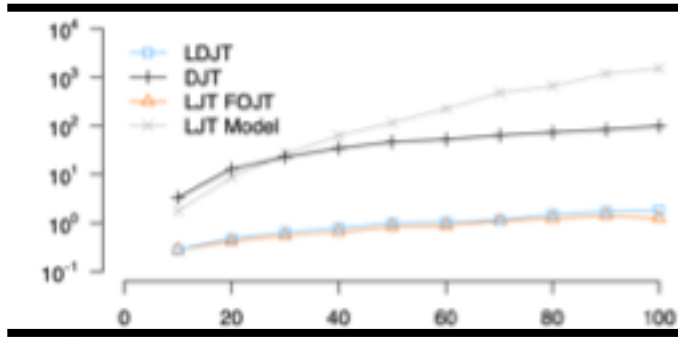


Fig. 3

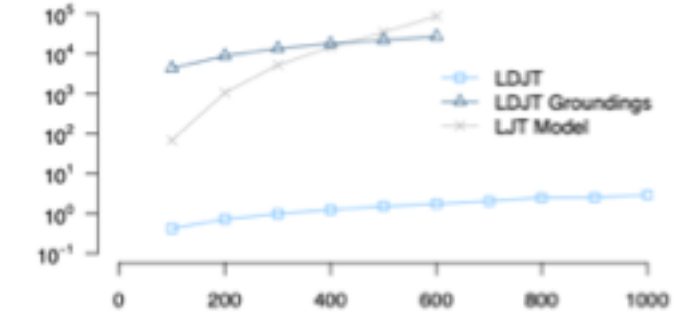
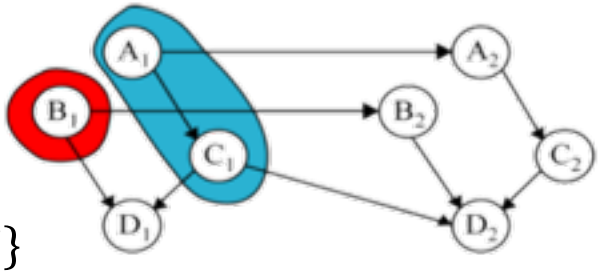


Fig. 4

A Note on Approximations

- Boyen-Koller algorithm
  - Assume independences in interface
    - Partition of interface required as input
      - E.g., $\{\{A, C\}, \{B\}\}$ in an interface $\{A, B, C\}$



  - Query no longer over complete interface but as a product over independent parts
    - Inter-slice message: Union of the results of queries of the individual parts of the partition
    - E.g., call LVE with the local model at $\boldsymbol{C}_{\tau}^{out}$ and a query for $\{A, C\}$ and a query for $\{B\}$:

$$\alpha_{\tau} = \{\mathrm{LVE}(G_{\tau}^{out}, \{A, C\}, \emptyset), \mathrm{LVE}(G_{\tau}^{out}, \{B\}, \emptyset)\}$$

  - Exact algorithm if independences actually hold, otherwise approximate

Xavier Boyen and Daphne Koller: Tractable Inference for Complex Stochastic Processes. In: *UAI-98 Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

A Note on Approximations

- Factored-frontier algorithm (Murphy, 2002)
  - Remember:
    - Probability propagation on polytree BNs: Send messages directly between the nodes
    - Probability propagation in general graphs called (loopy) belief propagation (BP): Send messages directly between the nodes
      - Similar to the colour passing scheme but sending actual messages instead of just colours
      - Exact if polytree, otherwise approximate
      - Lifted BP: Compress a graph (using colour passing) and send lifted messages on compressed graph
  - Core idea: Use BP over time
  - Lifted version for dynamic MLNs using lifted BP (Ahmadi et al, 2013)

Kevin P. Murphy: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley, 2002.
Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. In *Machine Learning*. 92(1):91-132, 2013.

Interim Summary

- Interfaces to separate past from present and present from future
  - Inter-slice messages
    - Forward message to propagate all information up to current slice to next slice
    - Backward message to propagate all information down to current slice to previous slice
- LDJT algorithm
  - LJT algorithm for intra-slice inference
  - Inter-slice messages to move in time
  - Algorithm-induced groundings possible but not necessarily preventable while keeping sequential separation up
  - Reduced complexity in terms of lifted interface

Outline: 6. Sequential Models & Inf.

A. *Lifted modelling of sequences*
   - Parameterised dynamic models (PDMs)
   - Modelling, semantics

B. *Lifted dynamic inference*
   - Inference tasks
   - Interfaces
   - Lifted dynamic junction tree algorithm (LDJT)
   - Theoretical analysis: complexity

C. *Keeping inference polynomial*
   - Problem of evidence over time in lifted models
   - Temporal approximate merging (TAMe)

The Problem with Evidence

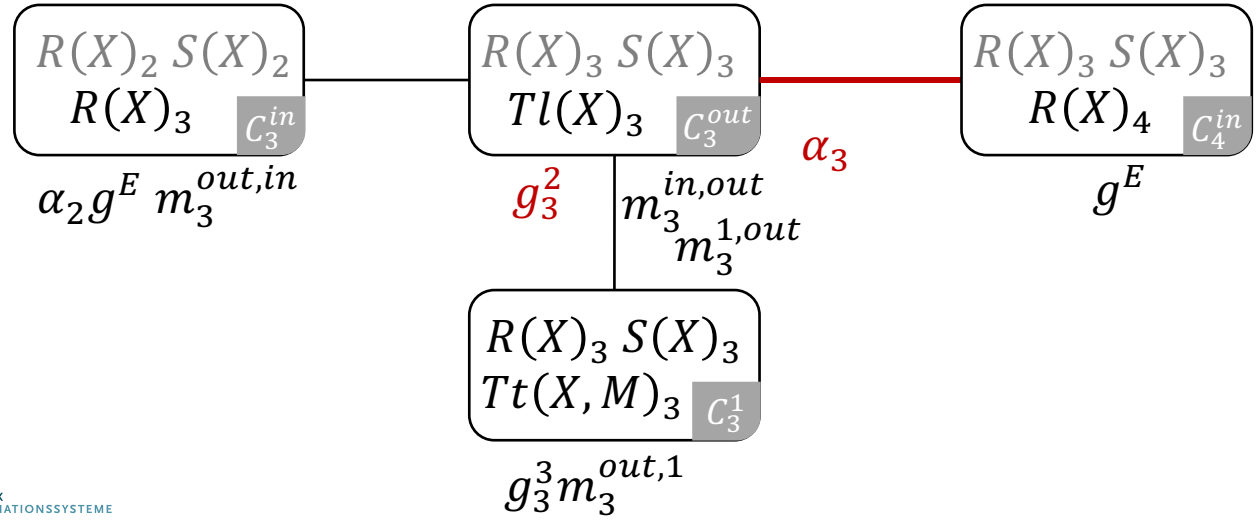
- Evidence can ground a model over time
- Non-symmetric evidence
  - Observe evidence for some instances in one time step
  - Observe evidence for a subset of these instances in another time step
  - Splits a logvar slowly over time
- Vanilla junction trees for each time step
  - Without any splits
- Forward message carries over splits, leading to slowly grounding a model over time

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Evidence over Time

- Slight variation of example
  - Replace $Epid_t$ with $R(X)_t$
- Evidence: $Tl_3(x_1) = true$
  - Split g_3^2 into
    - $g_3^{2=1}$ for x_1 and
    - $g_3^{2\neq 1}$ for $X \neq x_1$

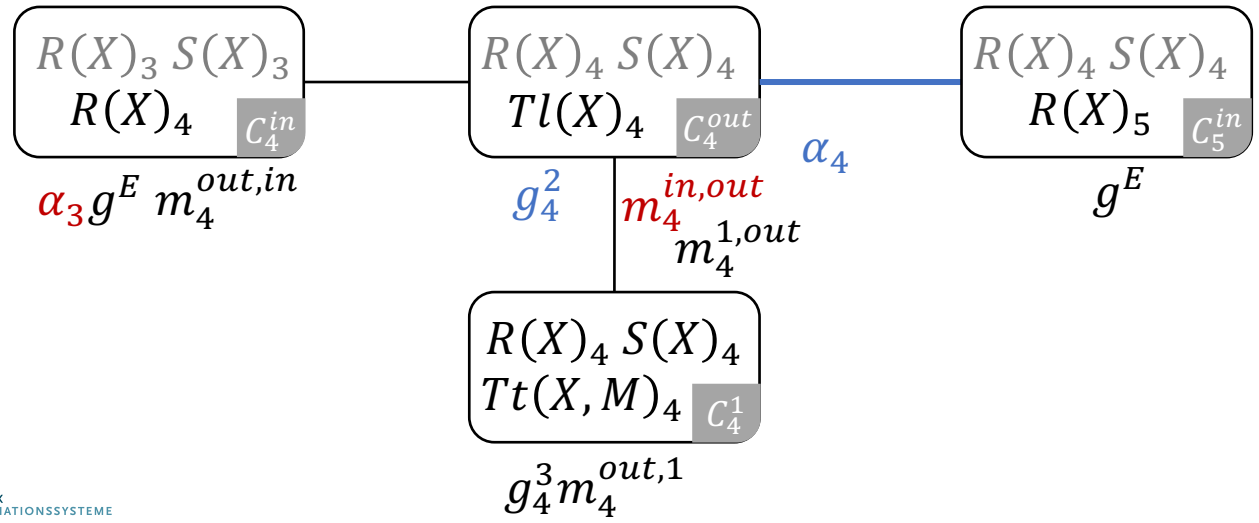
- α_3 consists of
  - $m_3^{in,out}$
  - $m_3^{1,out}$
  - $g_3^{2=1}$ and $g_3^{2\neq 1}$ with $Tl_3(X)$ eliminated



Evidence over Time

- Next step $\tau = 4$
- Evidence: $Tl_4(x_2) = true$
  - Split g_4^2 into
    - $g_4^{2=2}$ for x_2 and
    - $g_4^{2\neq 2}$ for $X \neq x_2$
- α_3 contains
  - $g_3^{2=1}$ and $g_3^{2\neq 1}$ with $Tl_3(X)$ eliminated
    - For $m_4^{in,out}$, X is split w.r.t. x_1
- In α_4, X is split w.r.t. x_1 and x_2

X is slowly grounded



Undoing Splits

- Need to undo splits to

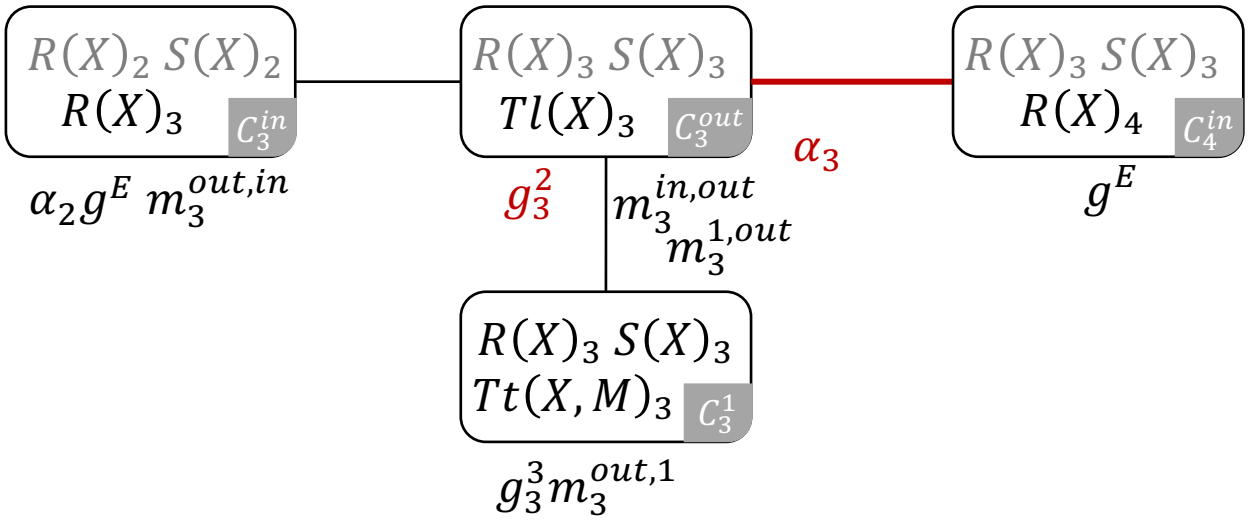
  keep reasoning polynomial w.r.t. domain sizes

1. Where can splits be undone efficiently?
2. How to undo splits?
3. Is it reasonable to undo splits?
   - Effect of slight differences in evidence?
   - Impact of evidence vs. temporal behaviour of model?

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

1. Where Can Splits Be Undone Efficiently?

- Evidence causes splits in a logical variable in the same way in all factors in a model
- LDJT always instantiates a vanilla junction tree
- Forward message carries over splits



Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

2. How to Undo Splits?

- The colour passing algorithm can efficiently identify exact symmetries
- But
  - Evidence causes differences in distributions

- Need to *find* approximate symmetries to undo splits caused by evidence
- Need a way to *merge* factors

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Comparing Parfactors: Approaches

- Comparing all marginals is expensive
- Comparing the joint distribution over the complete interface is expensive
- Comparing marginals of a *subset* of PRVs can determine non-similar factors similar
  - E.g.,
    - $\mathcal{D}(X) = \{x_1\}$

$R(X)$	$S(X)$	g
$false$	$false$	0
$false$	$true$	7
$true$	$false$	4
$true$	$true$	1

$R(X)$	$S(X)$	g
$false$	$false$	2
$false$	$true$	4
$true$	$false$	2
$true$	$true$	4

    - $P(S(x_1) = true)$: $\frac{2}{3}$ $\frac{2}{3}$
    - $P(R(x_1) = true)$: $\frac{5}{12}$ $\frac{1}{2}$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Comparing Parfactors

- Potentials determine distributions
- Similar ratios in potentials lead to similar marginals and similar factors
  - E.g.,
    - $\mathcal{D}(X) = \{x_1\}$

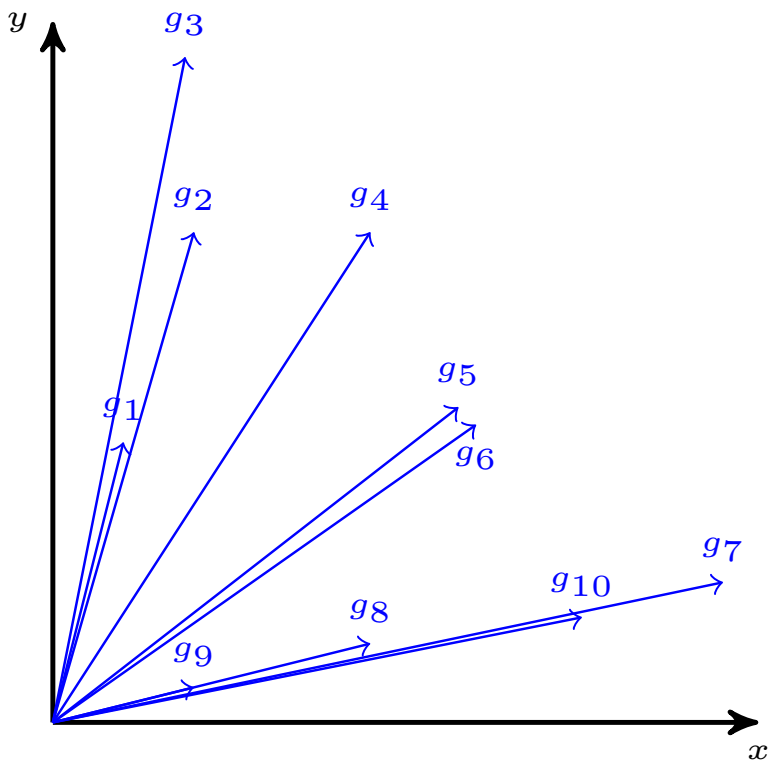
$R(X)$	$S(X)$	g
$false$	$false$	4
$false$	$true$	3
$true$	$false$	2
$true$	$true$	1

$R(X)$	$S(X)$	g
$false$	$false$	3.9
$false$	$true$	3.1
$true$	$false$	2.1
$true$	$true$	0.9

- $P(S(x_1) = true)$: $\frac{4}{10}$ $\frac{4}{10}$
- $P(R(x_1) = true)$: $\frac{3}{10}$ $\frac{3}{10}$
- $P(S(x_1) = true, R(x_1) = true)$: $\frac{1}{10}$ $\frac{0.9}{10}$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Identifying Similar Groups



- Groups are equal if they have the same full joint distribution
- Full joint distribution computationally hard to get

→ Use parfactors as vector

→ If vectors of two groups point in same direction, they have a similar full joint distribution

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Find Approximate Symmetries

- Cosine similarity for similarity of vectors

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- E.g.,

$Tl(X)$	$S(X)$	g
$false$	$false$	0
$false$	$true$	7
$true$	$false$	4
$true$	$true$	1

$Tl(X)$	$S(X)$	g
$false$	$false$	2
$false$	$true$	4
$true$	$false$	2
$true$	$true$	4

- $\cos(\theta) = \frac{0\cdot2+7\cdot4+4\cdot2+1\cdot4}{\sqrt{0+49+16+1}\cdot\sqrt{4+16+4+16}} \sim 0.7785$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Find Approximate Symmetries

- Cosine similarity for similarity of vectors

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- E.g.,

$Tl(X)$	$S(X)$	g
$false$	$false$	4
$false$	$true$	3
$true$	$false$	2
$true$	$true$	1

$Tl(X)$	$S(X)$	g
$false$	$false$	3.9
$false$	$true$	3.1
$true$	$false$	2.1
$true$	$true$	0.9

- $\cos(\theta) = \frac{4 \cdot 3.9 + 3 \cdot 3.1 + 2 \cdot 2.1 + 1 \cdot 0.9}{\sqrt{16+9+4+1} \cdot \sqrt{15.21+9.61+4.41+0.81}} \sim 0.9993$

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Find Approximate Symmetries

- Cosine similarity for similarity of vectors

$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- E.g.,

$Tl(X)$	$S(X)$	g
$false$	$false$	4
$false$	$true$	3
$true$	$false$	2
$true$	$true$	1

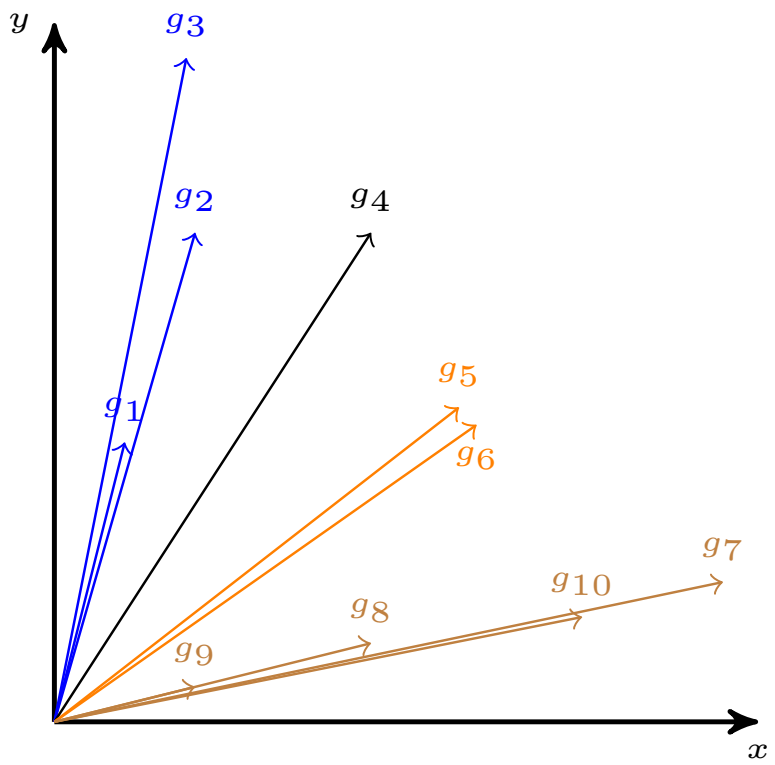
$Tl(X)$	$S(X)$	g
$false$	$false$	8
$false$	$true$	6
$true$	$false$	4
$true$	$true$	2

- $\cos(\theta) = \frac{4 \cdot 8 + 3 \cdot 6 + 2 \cdot 4 + 1 \cdot 3}{\sqrt{16+9+4+1} \cdot \sqrt{64+36+16+4}} = 1$

Use $1 - \cos(\theta)$ as distance function in clustering algorithm

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

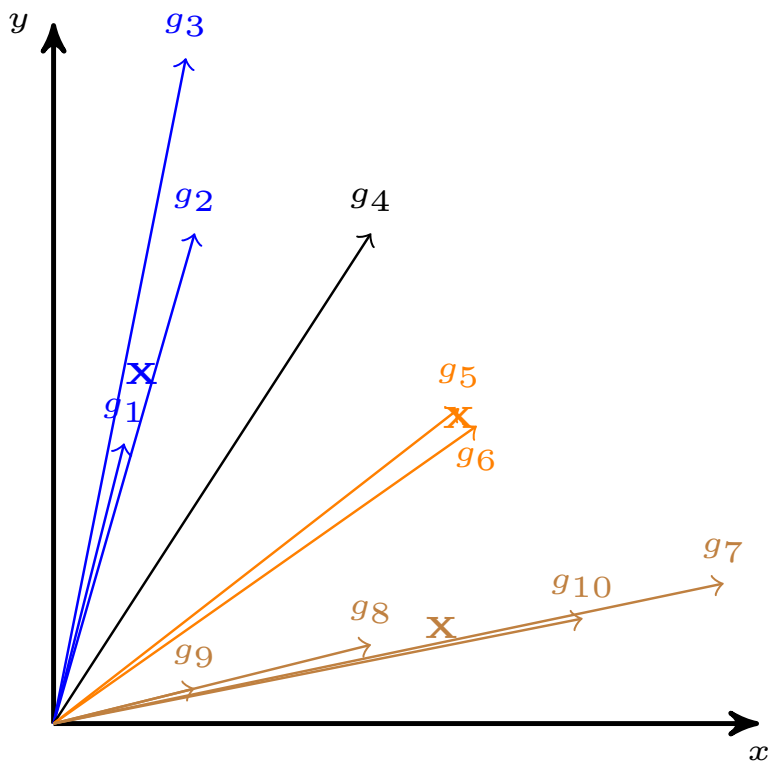
Cluster Groups



- Density-based clustering as unknown number of clusters
  - E.g., DBSCAN
- Cosine similarity as distance function
- Use ANOVA for testing fitness of clustering
  - Hypothesis testing
  - Do the cluster means sufficiently distinguish the clusters?

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Merge Groups



- Merge groups of cluster by calculating mean of cluster while accounting for groundings
- Replace old groups with merged group in forward message

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Merging Parfactors

$Tl(X)$	$S(X)$	g
$false$	$false$	4
$false$	$true$	3
$true$	$false$	2
$true$	$true$	1

$|\mathcal{D}(X)| = 4$

- Merge similar parfactors, *accounting for groundings*

$Tl(X')$	$S(X')$	g
$false$	$false$	7.9
$false$	$true$	6
$true$	$false$	3.9
$true$	$true$	2.1

$|\mathcal{D}(X')| = 4$

$Tl(X'')$	$S(X'')$	g
$false$	$false$	15.7
$false$	$true$	12.2
$true$	$false$	8.1
$true$	$true$	3.8

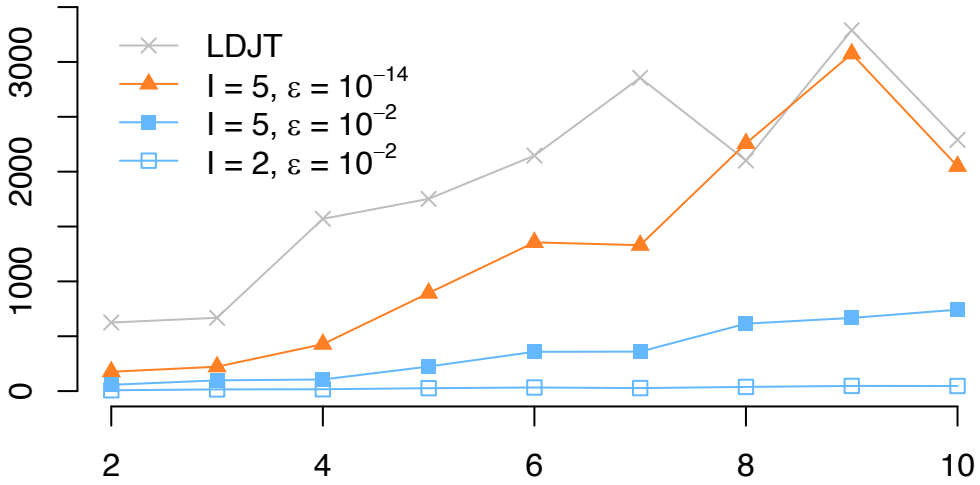
$|\mathcal{D}(X'')| = 2$

$|\mathcal{D}(X)| = 10$

$Tl(X)$	$S(X)$	g
$false$	$false$	$\frac{(4 \cdot 4 + 7.9 \cdot 4 + 15.7 \cdot 2)}{10} = 7.9$
$false$	$true$	$\frac{(3 \cdot 4 + 6 \cdot 4 + 12.2 \cdot 2)}{10} = 6.04$
$true$	$false$	$\frac{(2 \cdot 4 + 3.9 \cdot 4 + 8.1 \cdot 2)}{10} = 3.98$
$true$	$true$	$\frac{(1 \cdot 4 + 2.1 \cdot 4 + 3.8 \cdot 2)}{10} = 2$

Runtimes and Error

- DBSCAN for Clustering
- ANOVA for checking fitness of clusters

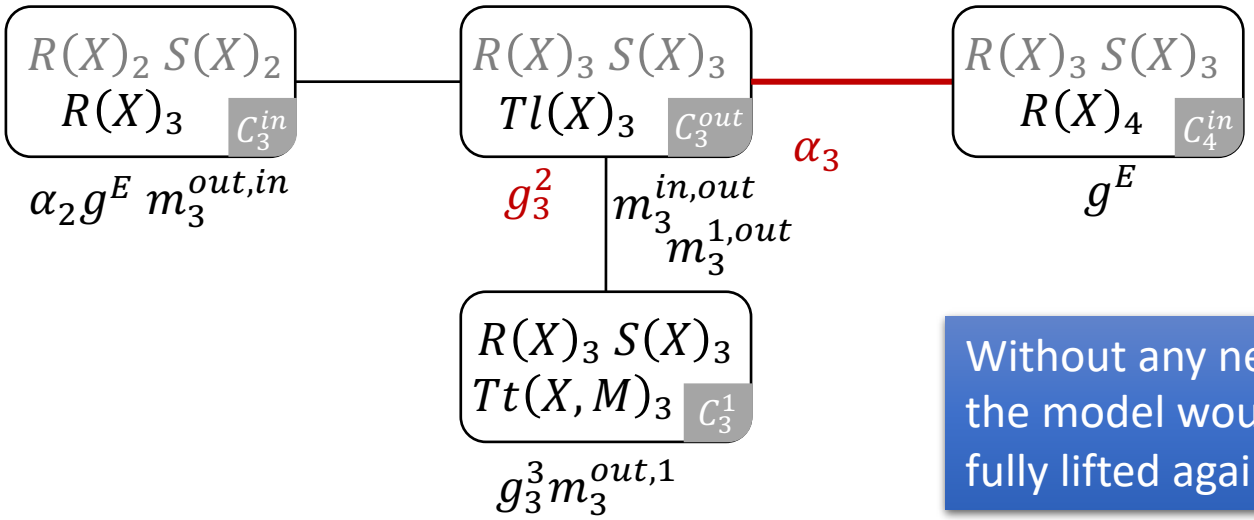


π	Max	Min	Average
0	0.0001537746121	0.0000000001720	0.0000191206488
2	0.0000000851654	0.0000000000001	0.0000000111949
4	0.0000000000478	0	0.0000000000068

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Is It Reasonable to Undo Splits?

- *Approximate* forward message
- For each time step, sequential behaviour is multiplied onto the forward message
- Indefinitely bounded error due to sequential behaviour



Without any new evidence, the model would become fully lifted again!

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Interim Summary

- Need to undo splits to

  keep reasoning polynomial w.r.t. domain sizes

1. Where can splits be undone efficiently?
   - Undo splits in a forward message
2. How to undo splits?
   - Find approximate symmetries
   - Merge based on groundings
3. Is it reasonable to undo splits?
   - Yes, due to the temporal model behaviour (indefinitely bounded error)
     - Achieve fully lifted model again without new evidence

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *Proceedings of the ECAI 2020*, 2020.

Outlook

- Changing domains over time/sequences
  - Towards open-world assumption
- Non-stationarity
  - Outside forces change the distributions
  - C.f., reinforcement learning
- Markov-k
  - Influence might be further in the future than just the next step → Something that will be even more apparent during next topic, decision making, where actions might only have an effect after c steps
- OngoingResearch@IFIS

Outline: 6. Sequential Models & Inf.

A. *Lifted modelling of sequences*
   - Parameterised dynamic models (PDMs)
   - Modelling, semantics

B. *Lifted dynamic inference*
   - Inference tasks
   - Interfaces
   - Lifted dynamic junction tree algorithm (LDJT)
   - Theoretical analysis: complexity

C. *Keeping inference polynomial*
   - Problem of evidence over time in lifted models
   - Temporal approximate merging (TAMe)

⇒ Next: Decision Making