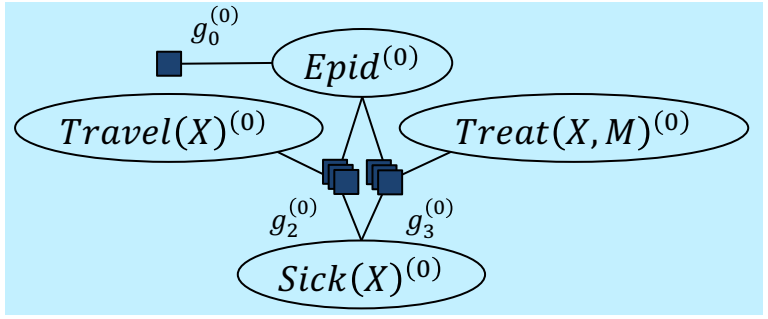
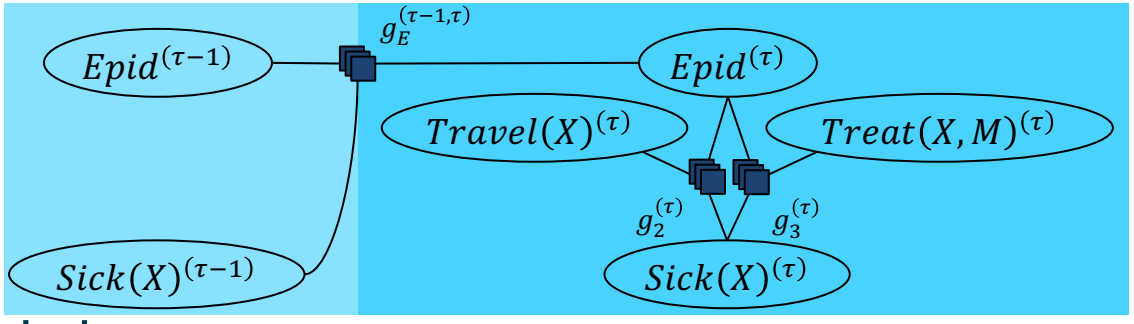
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Dynamic Probabilistic Relational Models

Lifted Sequential Models and Inference





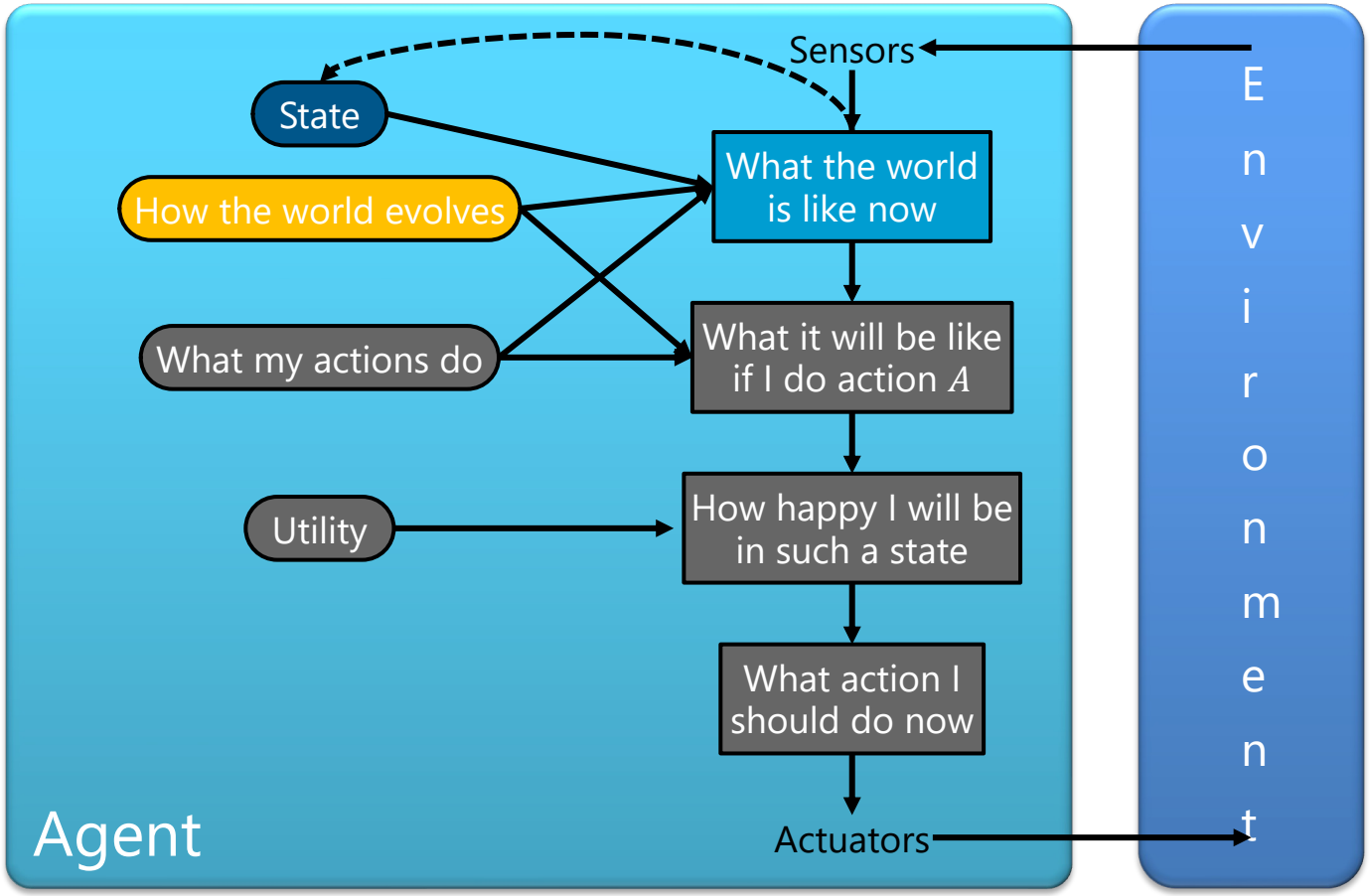
Marcel Gehrke

Contents

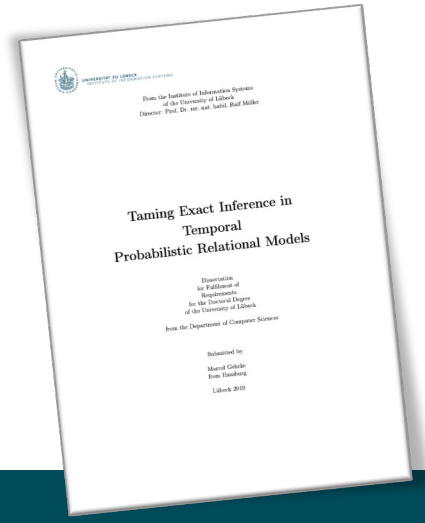
1. **Introduction**
   - StaRAI: Agent, context, motivation
2. **Foundations**
   - Logic
   - Probability theory
   - Probabilistic graphical models (PGMs)
3. **Probabilistic Relational Models (PRMs)**
   - Parfactor models, Markov logic networks
   - Semantics, inference tasks
4. **Exact Lifted Inference**
   - Lifted Variable Elimination
   - Lifted Junction Tree Algorithm
   - First-Order Knowledge Compilation
5. **Lifted Sequential Models and Inference**
   - Parameterised models
   - Semantics, inference tasks, algorithm
6. **Lifted Decision Making**
   - Preferences, utility
   - Decision-theoretic models, tasks, algorithm
7. **Approximate Lifted Inference**
8. **Lifted Learning**
   - Parameter learning
   - Relation learning
   - Approximating symmetries

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

- Further topics
  3. (Episodic) PRMs
  4. Lifted inference (in episodic PRMs)
  5. Lifted sequential PRMs and inference
  6. Lifted decision making



Main source:
Taming Exact Inference in Temporal Probabilistic Relational Models
Marcel Gehrke, PhD thesis, 2022
https://www.ifis.uni-luebeck.de/uploads/tx_wapublications/Diss_Gehrke_public_02.pdf


Taming Exact Inference in Temporal Probabilistic Relational Models

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)

Outline: 6. Lifted Sequential Models & Inference

A. Lifted modelling of sequences

- Templates, modelling
- Semantics, inference tasks

B. Lifted dynamic inference

- Interfaces, lifted dynamic junction tree algorithm (LDJT)
- Theoretical analysis: complexity
- Keeping inference polynomial: Temporal approximate merging (TAMe)

System State over Time

- Set of random variables to describes a system's state
  - $\boldsymbol{R} = \{R_1, \dots, R_n\}$
  - Template variables
- System state at time step t:
  - Characterised by an assignment (compound event) to $\boldsymbol{R}$ at time step t
    - Instantiate template variables with t, denoted by $\boldsymbol{R}^{(t)} = \left\{R_1^{(t)}, \dots, R_n^{(t)}\right\}$
    - Assignment with values from $\mathrm{ran}(R_i)$: $\boldsymbol{r}^{(t)} = \left\{R_1^{(t)} = r_1^{(t)}, \dots, R_n^{(t)} = r_n^{(t)}\right\}, r_i^{(t)} \in \mathrm{ran}(R_i)$
- System state over a discrete interval $[t_1, t_2], t_1 < t_2$
  - Set of random variables: $\boldsymbol{R}^{(t_1:t_2)} = \left\{\boldsymbol{R}^{(t)} \middle| t \in [t_1, t_2]\right\}$
  - State "sequence": $\boldsymbol{r}^{(t_1:t_2)} = \left\{\boldsymbol{r}^{(t)} \middle| t \in [t_1, t_2]\right\}$
    - As actual sequence: $\boldsymbol{r}^{(t_1:t_2)} = \left(\boldsymbol{r}^{(t)} \middle| t \in [t_1, t_2]\right)$

System State over Time

- Given a set of template variables $\boldsymbol{R} = \{R_1, \dots, R_n\}$ and an end point T
  - Set of random variables: $\boldsymbol{R}^{(0:T)}$
- One possible world: compound event $\boldsymbol{r}^{(0:T)}$ of $\boldsymbol{R}^{(0:T)}$ with probability assigned
  - Called trajectory
- Full joint probability distribution over all possible trajectories

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{R}^{(0:T)}\right) = P\left(\boldsymbol{R}^{(0)}, \dots, \boldsymbol{R}^{(T)}\right) = P\left(R_1^{(0)}, \dots, R_n^{(0)}, \dots, R_1^{(T)}, \dots, R_n^{(T)}\right)$$

  - Apply chain rule:

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{R}^{(0:T)}\right) = P\left(\boldsymbol{R}^{(0)}\right) \prod_{t=1}^{T} P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{(0:(t-1))}\right)$$

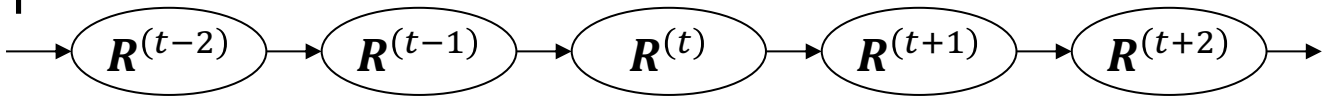
  - Very complex distribution without simplifying assumptions

Simplifying Assumption 1: Markov Assumption

- Markov Assumption: Next state depends only on current state
  - Formal: $\left(\boldsymbol{R}^{(t+1)} \perp \boldsymbol{R}^{(0:(t-1))} \middle| \boldsymbol{R}^{(t)}\right)$
- Effect on $P\left(\boldsymbol{R}^{(0:T)}\right)$: Factorisation over t

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{R}^{(0:T)}\right) = P\left(\boldsymbol{R}^{(0)}\right) \prod_{t=1}^{T} P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{(t-1)}\right)$$

  - Also called *first-order Markov process*
  - Partitioning of the model into (time) slices
  - Graphical representation:

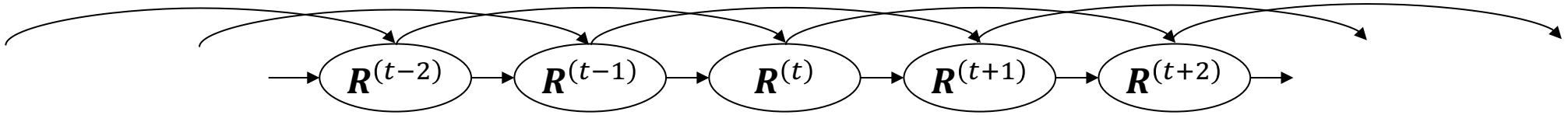


Simplifying Assumption 1: Markov Assumption

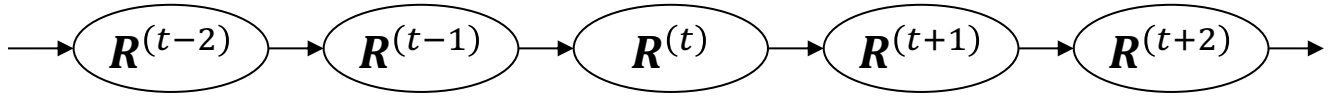
- Generalisation of the Markov assumption: Markov-k
  - Next state depends on k previous states:

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{R}^{(0:T)}\right) = P\left(\boldsymbol{R}^{(0)}\right) \prod_{t=1}^{k} P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{(0:(t-1))}\right) \prod_{t=k+1}^{T} P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{((t-k):(t-1))}\right)$$

  - Graphical representation for $k = 2$, called *second-order Markov process*



  - Markov assumption $\triangleq$ Markov-k with $k = 1$



Simplifying Assumption 2: Stationary System

- Given a set of template variables $\boldsymbol{R} = \{R_1, \dots, R_n\}$ and an end point T
- Full joint probability distribution over all possible trajectories
  - With Markov assumption: Factorisation over t

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{R}^{(0:T)}\right) = P\left(\boldsymbol{R}^{(0)}\right) \prod_{t=1}^{T} P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{(t-1)}\right)$$

- Assumption Stationarity: $P\left(\boldsymbol{R}^{(t)} \middle| \boldsymbol{R}^{(t-1)}\right)$ identical for each t
  - Formally, there exists a *transition model* $P(\boldsymbol{R}' | \boldsymbol{R})$ such that for each $t \geq 1$ it holds that

$$P\left(\boldsymbol{R}^{(t)} = \boldsymbol{r}' \middle| \boldsymbol{R}^{(t-1)} = \boldsymbol{r}\right) = P(\boldsymbol{R}' = \boldsymbol{r}' | \boldsymbol{R} = \boldsymbol{r})$$

For a stationary system with Markov assumption, it is sufficient to specify the following distributions: $P\left(\boldsymbol{R}^{(0)}\right), P\left(\boldsymbol{R}^{(\tau)} \middle| \boldsymbol{R}^{(\tau-1)}\right)$

What is a problem with that?

Observable and Latent Variables

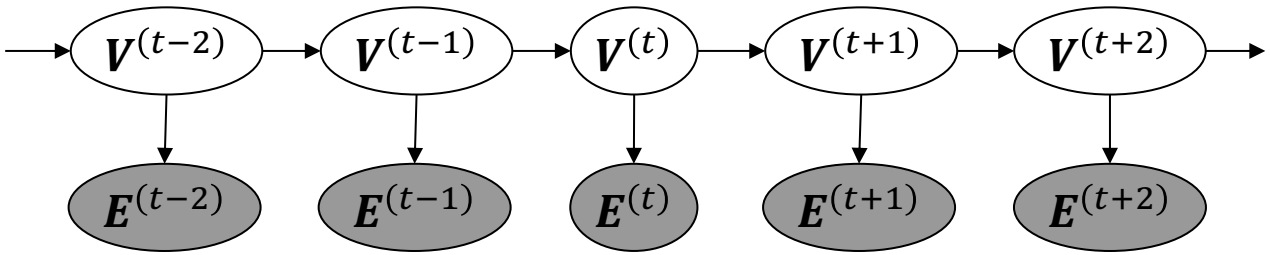
- Set of template variables $\boldsymbol{R}$ usually partitioned into evidence variables $\boldsymbol{E}$ (observations available in each step) and latent variables $\boldsymbol{V}$
- Full joint can then be factorised as follows:

$$P_{\boldsymbol{R}}^{T} = P\left(\boldsymbol{V}^{(0:T)}, \boldsymbol{E}^{(0:T)}\right) = P\left(\boldsymbol{V}^{(0)}\right) \prod_{t=1}^{T} P\left(\boldsymbol{V}^{(t)} \middle| \boldsymbol{V}^{(t-1)}\right) P\left(\boldsymbol{E}^{(t)} \middle| \boldsymbol{V}^{(t)}\right)$$

Transition model

Sensor model

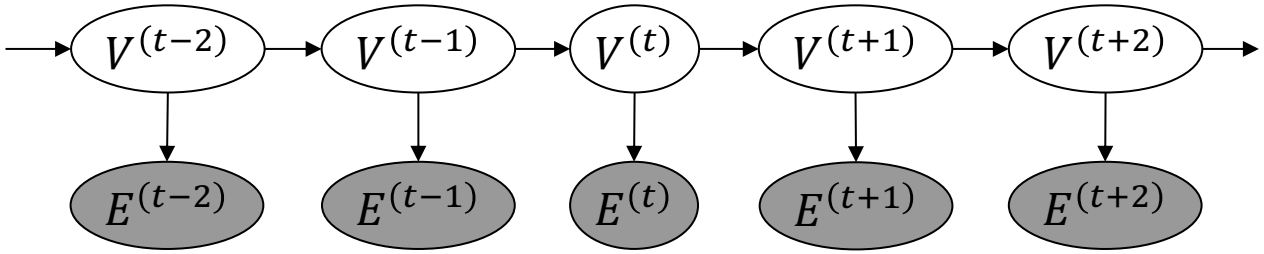
What (conditional) independences hold?



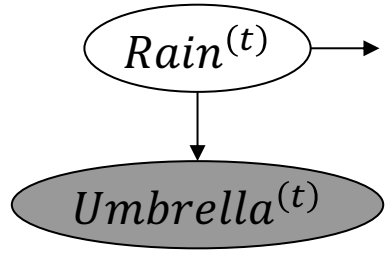
Hidden Markov Modell (HMM)

- Simplest sequential model with $\boldsymbol{R} = \{V, E\}$
  - $\boldsymbol{V} = \{V\}, \boldsymbol{E} = \{E\}$
    - Latent variable V
    - Evidence variable E
  - Model:
    - $P(V^{(0)})$ prior information on initial state
    - $\{P(V^{(\tau)}|V^{(\tau-1)}), P(E^{(\tau)}|V^{(\tau)})\}$ transition / sensor model
  - Full joint: $P_{V,E}^{T} = P(V^{(0)}) \prod_{t=1}^{T} P(V^{(t)}|V^{(t-1)}) P(E^{(t)}|V^{(t)})$

HMM can be found in various application scenarios such as
- Signal processing
- Bio informatics
- Text understanding



$P(r^{(0)})$
0.5

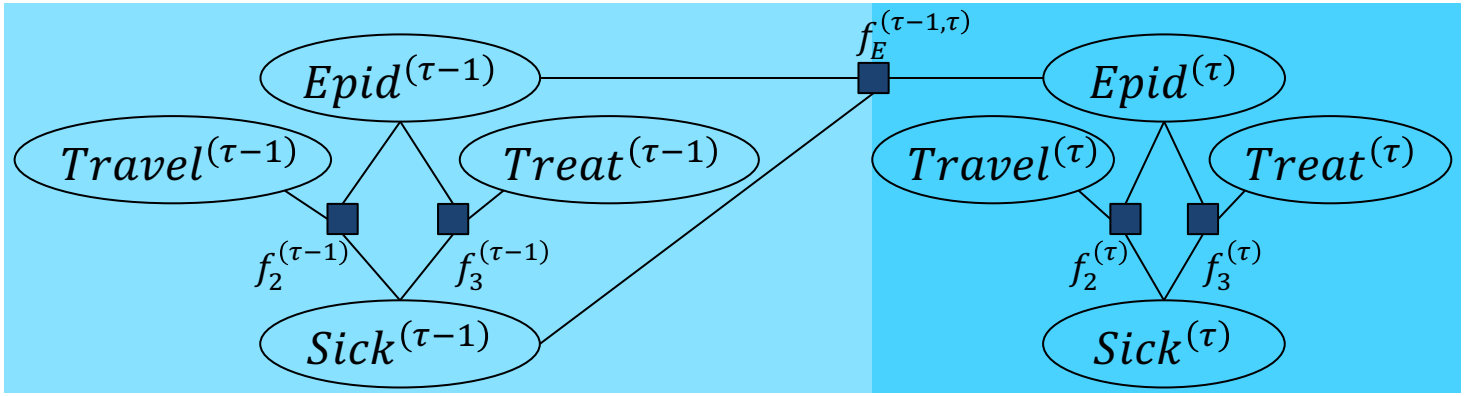


$R^{(t-1)}$	$P(r^{(t)}\|R^{(t-1)})$
true	0.7
false	0.3

$R^{(t)}$	$P(u^{(t)}\|R^{(t)})$
true	0.9
false	0.2

General Factorisation in Sequential Models

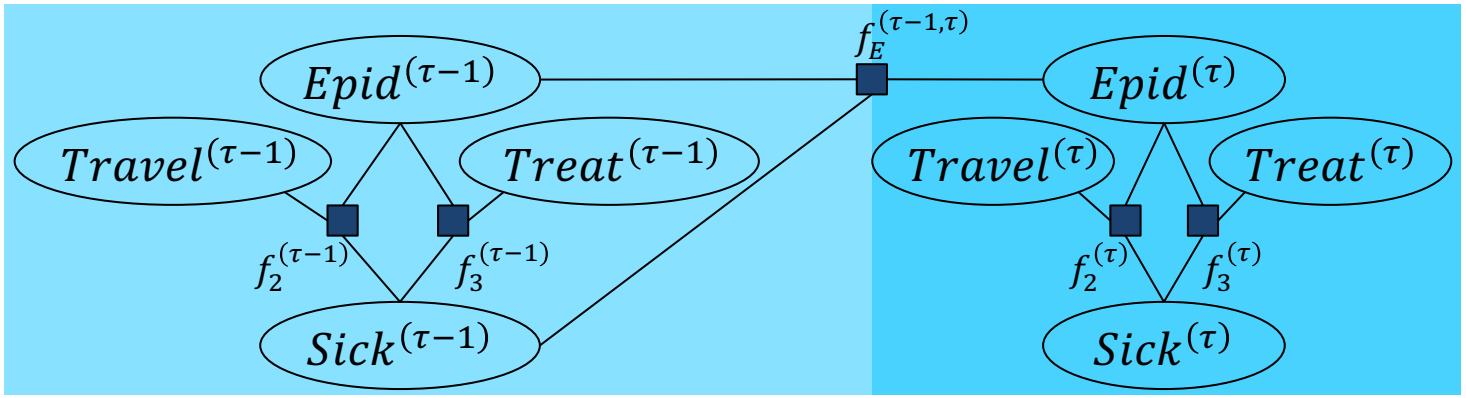
- Transition from one step to the next triggered by new observation / execution of an action
  - Markov assumption: Effect of action in current or next step
- Formalisation
  - *Intra-slice f*actors: Describe behaviour / effects within a step
    - Factor arguments only from step τ
      - E.g., g_2: $Epid^{(\tau)}$, $Sick^{(\tau)}$, $Travel^{(\tau)}$
  - *Inter-slice* factors: Describe transitions / behaviour with effects in next step
    - Factor arguments from $\tau - 1, \tau$
      - E.g., f_E: $Epid^{(\tau-1)}$, $Sick^{(\tau-1)}$, $Epid^{(\tau)}$



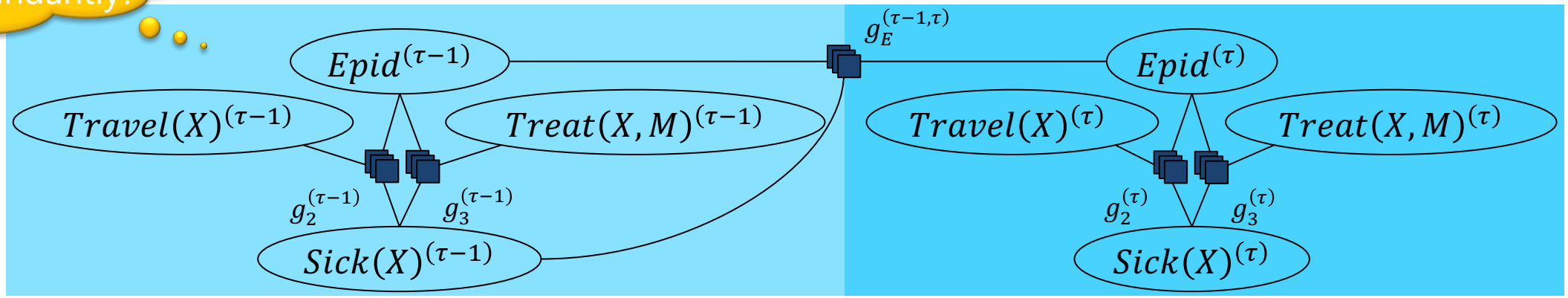
Lifting the Factorisation

- Assume regularities / symmetries among factors due to relational setup

→ Use logical variables to encode symmetries compactly



What information is depicted redundantly?



2-slice Model & 1.5-slice Model

Partial specification of a sequential model (without $P(\boldsymbol{R}^{(0)})$)

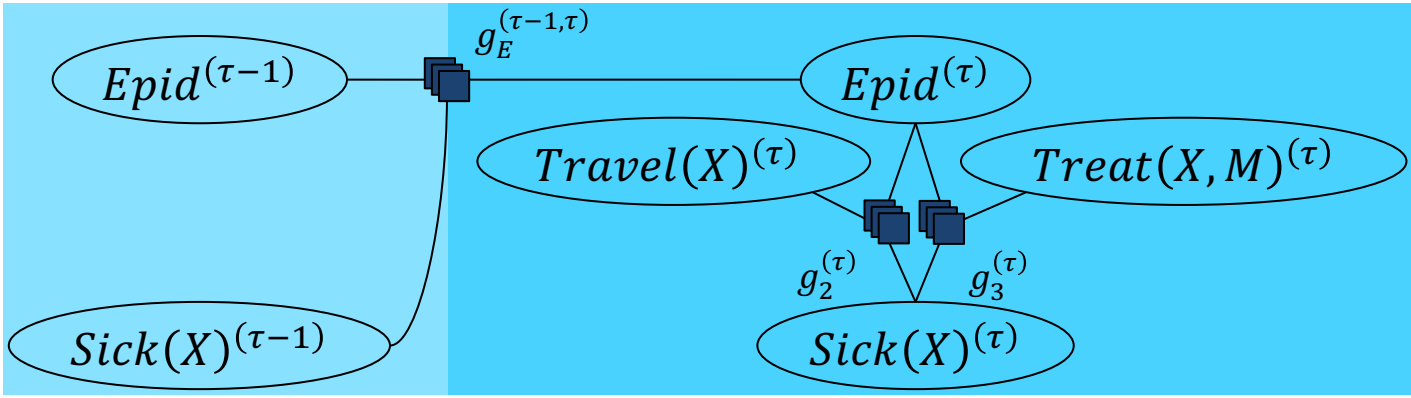
- 2-slice model (in effect only a 1.5-slice model)
  - Specification of a transition model $P(\boldsymbol{R}^{(\tau)} | \boldsymbol{R}^{(\tau-1)})$ based on a factorisation
    - Template model that gets instantiated by replacing τ with a (time) step t
    - Sufficient: Intra-slice description for τ, Inter-slice description from $\tau - 1$ to τ
  - For parfactor-based models: $G^{\rightarrow} = \left\{g_i^{(\tau)}\right\}_{i=1}^{n} \cup \left\{g_j^{(\tau-1,\tau)}\right\}_{j=1}^{m}$ with
    - Intra-slice parfactors

$$g_i^{(\tau)} = \phi_i^{(\tau)}\left(A_1^{(\tau)}, \ldots, A_{k_i}^{(\tau)}\right)_{|C_i^{(\tau)}}$$

    - Inter-slice parfactors

$$g_j^{(\tau-1,\tau)} = \phi_j^{(\tau-1,\tau)}\left(A_1^{(\nu)}, \ldots, A_{k_j}^{(\nu)}\right)_{|C_j^{(\tau-1,\tau)}}$$

      - $\nu \in \{\tau, \tau - 1\}$



Sequential Models

- Assumptions: Markov-1, stationary process

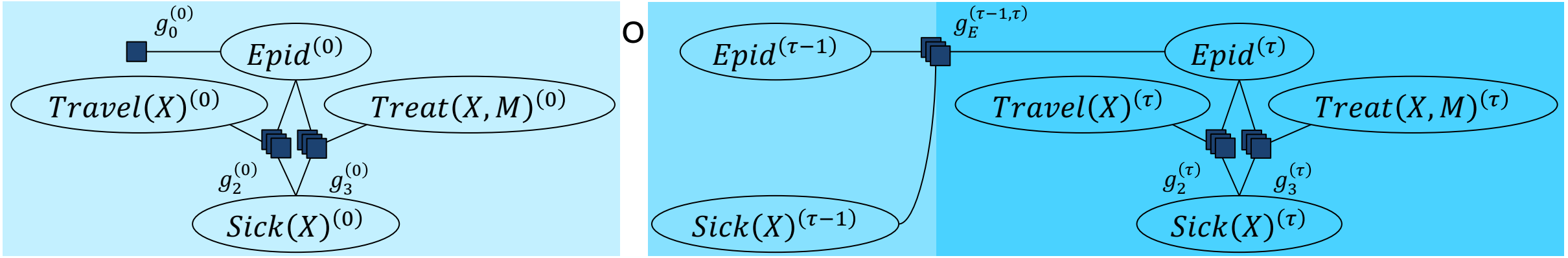
- Typical definition: Sequential model is a tuple $(M^0, M^{\rightarrow})$, where
  - M^0 describes the behaviour at step 0 (only intra-slice behaviour as no previous step)
  - $M^{\rightarrow}$ is a 2-slice model describing intra- and inter-slice behaviour
  - Existing model types: dynamic BNs, HMMs, dynamic parfactor models, dynamic MLNs, …
- Frequent connection between $M^0, M^{\rightarrow}$
  - In factor-based models: Intra-slice behaviour in $M^{\rightarrow}$ for step τ equal to behaviour in M^0, i.e., intra-slice parfactors in $M^{\rightarrow}$ with $\tau = 0$
  - M^0 may also contain prior information encoded in distributions
    - In BN-based models: prior distributions in M^0

Dynamic Parfactor-based Model

- Dynamic parfactor-based model $(G^0, G^{\rightarrow})$ with
  - $G^0 = \left\{g_i^{(0)}\right\}_{i=1}^{n_0}, g_i^{(0)} = \phi_i^{(0)}\left(A_1^{(0)}, \ldots, A_{k_i}^{(0)}\right)$
  - $G^{\rightarrow}$ a 2-slice model, i.e., $G^{\rightarrow} = \left\{g_i^{(\tau)}\right\}_{i=1}^{n} \cup \left\{g_j^{(\tau-1,\tau)}\right\}_{j=1}^{m}$
  - Example: $G^0 = \left\{g_2^{(0)}, g_3^{(0)}\right\} \cup \left\{g_0^{(0)}\right\}$, $G^{\rightarrow} = \left\{g_2^{(\tau)}, g_3^{(\tau)}\right\} \cup \left\{g_E^{(\tau-1,\tau)}\right\}$

G^0 often consists of intra-slice parfactors of $G^{\rightarrow}$ with $\tau = 0$, possibly extended by pseudo prior distributions, i.e.,

$$\left\{g_i^{(0)}\right\}_{i=1}^{n_0} = \left\{g_i^{(\tau)|\tau=0}\right\}_{i=1}^{n} \cup \left\{g_i^{(0)}\right\}_{i=1}^{n'}$$



Semantics

- Semantics defined over the full joint probability distribution given an end point $T > 0$
  - Unroll dynamic model $M = (M^0, M^{\rightarrow})$ for T steps

$$M^{(0:T)} = M^0 \cup \bigcup_{t=1}^{T} M^{\rightarrow|\tau=t}$$

    - $M^{\rightarrow|\tau=t}$: Replace τ with t in $M_{\rightarrow}$
  - Yields an episodic model with known semantics, i.e., a model which represents a full joint probability distribution as a (normalised) product of local distributions

$$P_M^T = \frac{1}{Z} \prod_{\zeta \in gr\left(M^{(0:T)}\right)} \zeta$$

    - ζ a grounded parfactor, if $M^0, M^{\rightarrow}$ parfactor-based, or
      ζ a CPD, if $M^0, M^{\rightarrow}$ BNs (then $Z = 1$), or
      ζ a grounded potential function if $M^0, M^{\rightarrow}$ MLNs

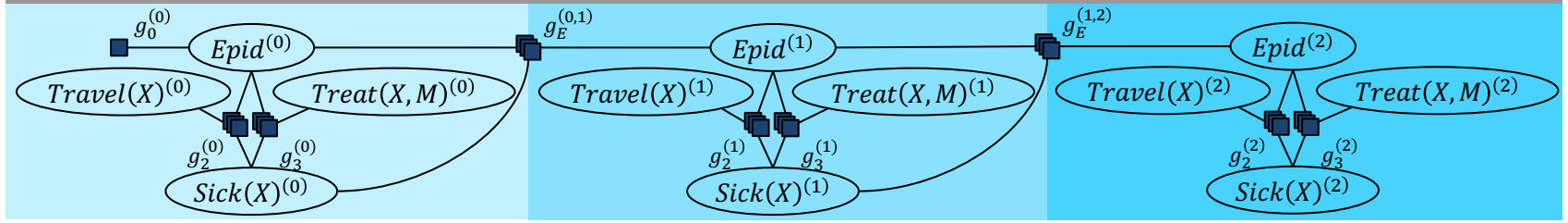
Semantics: Example – Unrolling

- Dynamic model
  $G = (G^0, G^{\rightarrow})$
- Unrolled model for $T = 2$

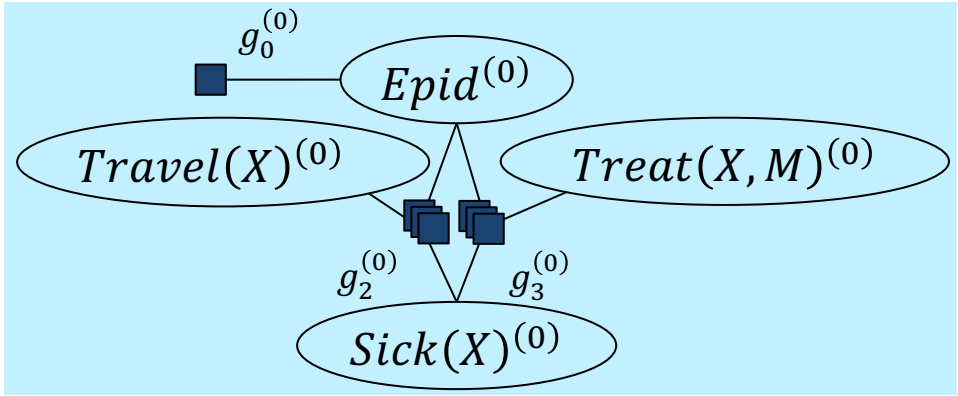
$G^{(0:2)}$

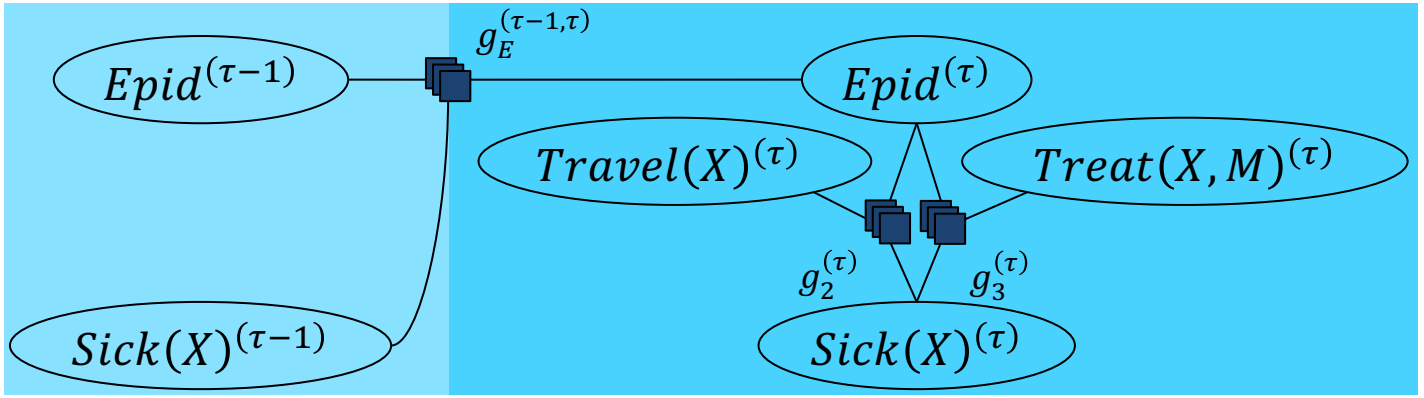
$$= G^0 \cup \bigcup_{t=1}^{2} G^{\rightarrow|\tau=t}$$

$$G = \left\{g_0^{(0)}, g_2^{(0)}, g_3^{(0)}\right\} \cup \left\{g_E^{(0,1)}\right\} \cup \left\{g_2^{(1)}, g_3^{(1)}\right\} \cup \left\{g_E^{(1,2)}\right\} \cup \left\{g_2^{(2)}, g_3^{(2)}\right\}$$



$t = 0$ $t = 1$ $t = 2$



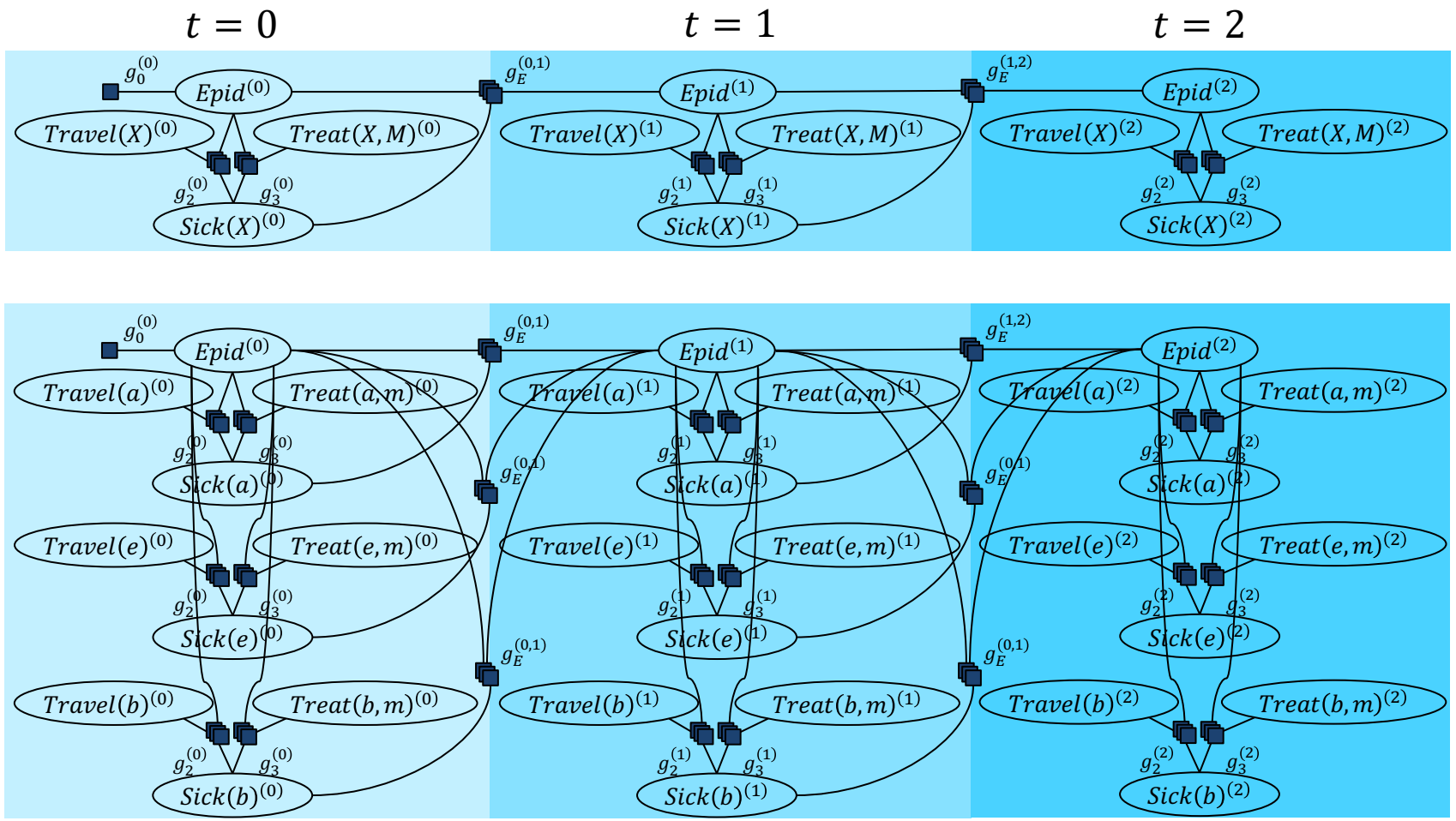


Semantics: Example – Grounding

- Dynamic model
  $G = (G^0, G^{\rightarrow})$
- Unrolled model for $T = 2$

$$G^{(0:2)} = G^0 \cup \bigcup_{t=1}^{2} G^{\rightarrow|\tau=t}$$

- Grounding
  $F = gr\left(G^{(0:2)}\right)$
  - Domains
    - $\mathrm{dom}(X) = \{a, e, b\}$
    - $\mathrm{dom}(M) = \{m\}$



Semantics: Example – Full Joint

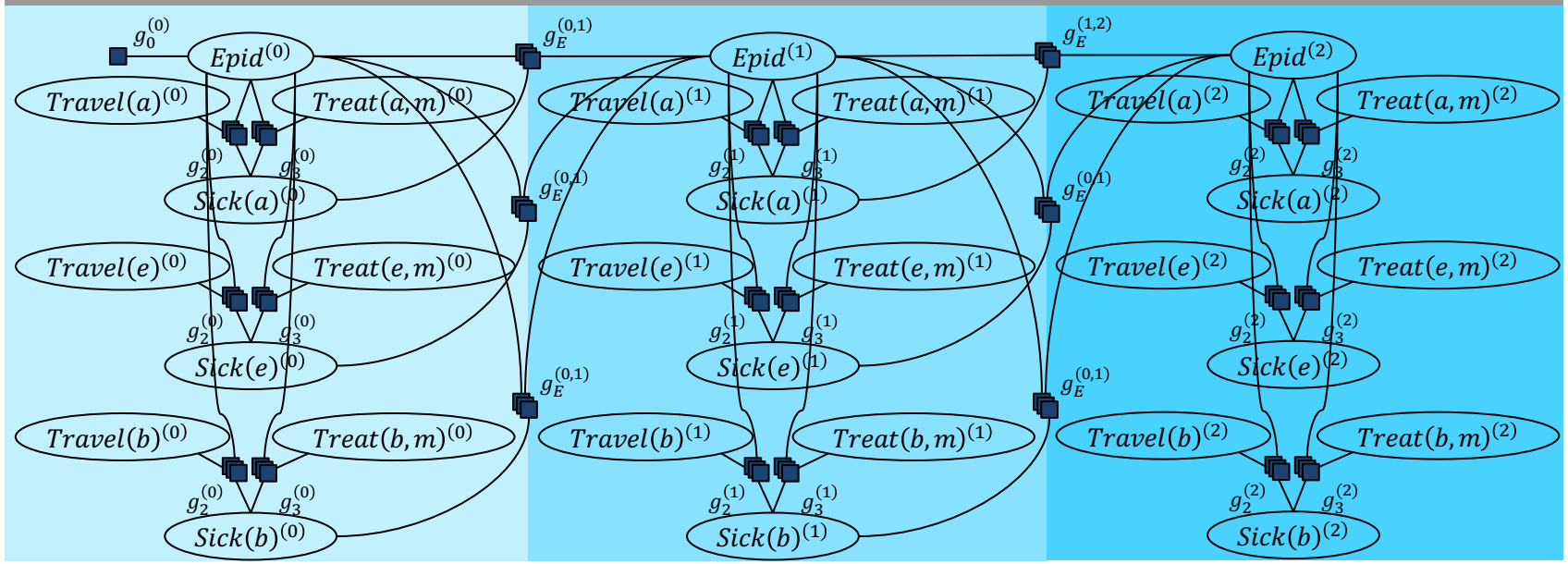
- Dynamic model
  $G = (G^0, G^{\rightarrow})$
- Unrolled model for $T = 2$

$$G^{(0:2)} = G^0 \cup \bigcup_{t=1}^{2} G^{\rightarrow|\tau=t}$$

- Grounding
  $F = gr\left(G^{(0:2)}\right)$
- Full joint

$$P_G^2 = \frac{1}{Z}\prod_{f\in F} f$$

$$P_G^2 = \frac{1}{Z}\prod_{f\in F} f = \frac{1}{Z}\cdot f_{2,a}^{(0)}\cdot f_{3,a}^{(0)}\cdot f_{2,e}^{(0)}\cdot f_{3,e}^{(0)}\cdot f_{2,b}^{(0)}\cdot f_{3,b}^{(0)}\cdot f_0^{(0)}$$
$$\cdot f_{2,a}^{(1)}\cdot f_{3,a}^{(1)}\cdot f_{2,e}^{(1)}\cdot f_{3,e}^{(1)}\cdot f_{2,b}^{(1)}\cdot f_{3,b}^{(1)}\cdot f_{E,a}^{(0,,1)}\cdot f_{E,e}^{(0,,1)}\cdot f_{E,b}^{(0,,1)}$$
$$\cdot f_{2,a}^{(2)}\cdot f_{3,a}^{(2)}\cdot f_{2,e}^{(2)}\cdot f_{3,e}^{(2)}\cdot f_{2,b}^{(2)}\cdot f_{3,b}^{(2)}\cdot f_{E,a}^{(1,2)}\cdot f_{E,e}^{(1,2)}\cdot f_{E,b}^{(1,2)}$$



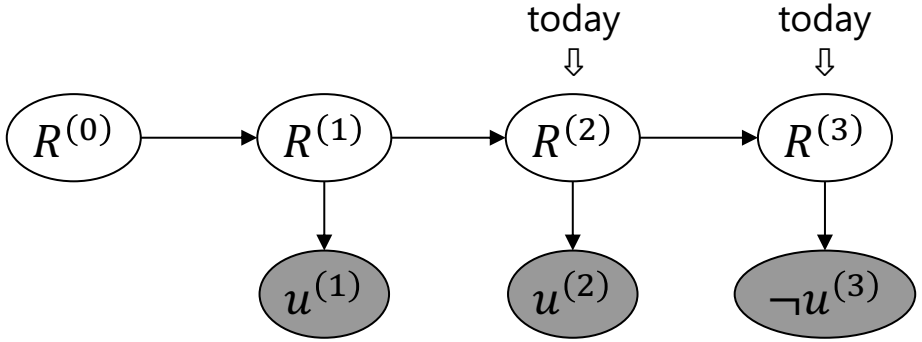
Query Answering Problem & Query Types

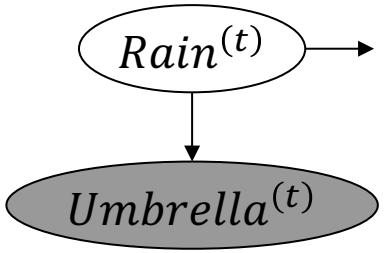
- Filtering
  - What is the probability that it rains today ($t = 2$) given all observations about umbrellas up until today (t)?

$$P\left(r^{(2)} \middle| u^{(1)}, u^{(2)}\right)$$

  - Next day (t ← $t + 1$): What is the probability that it rains today ($t = 3$) given all observations about umbrellas up until today (t)?

$$P\left(R^{(3)} \middle| u^{(1)}, u^{(2)}, \neg u^{(3)}\right)$$

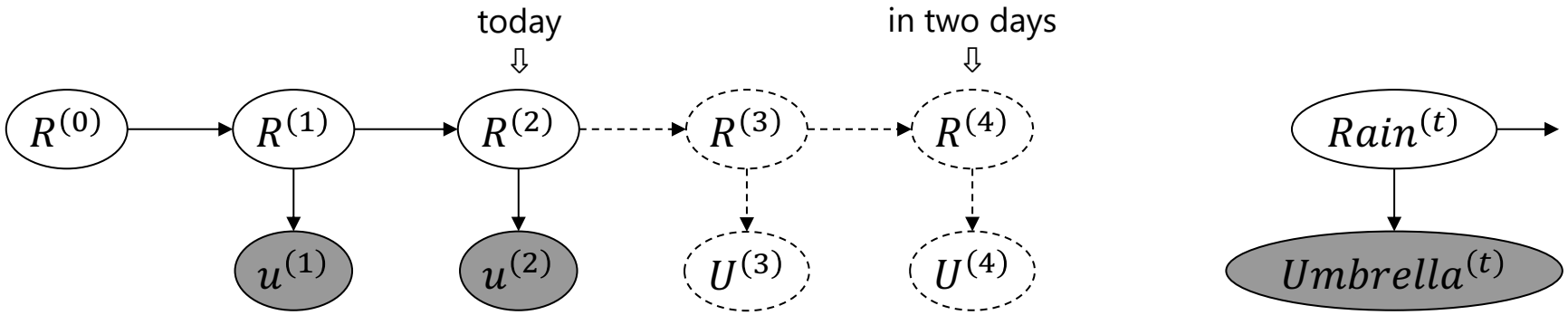




Query Answering Problem & Query Types

- Prediction
  - What is the probability that it rains in two days ($t + 2$) given all observations about umbrellas up until today ($t = 2$)?

$$P\left(R^{(4)} \middle| u^{(1)}, u^{(2)}\right)$$

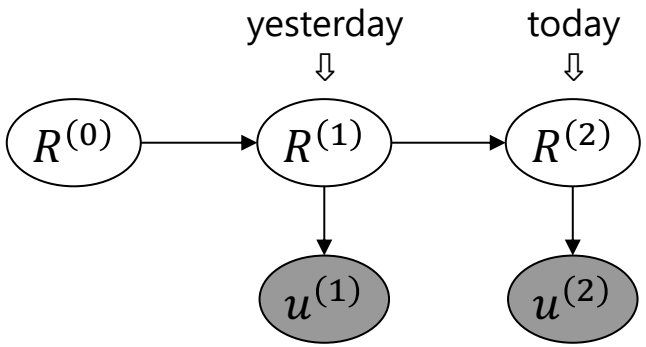


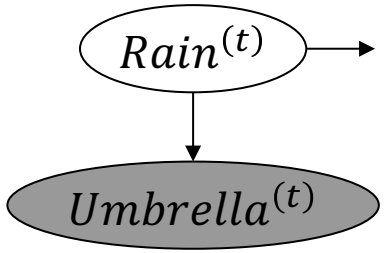
Query Answering Problem & Query Types

- Hindsight
  - What is the probability that is rained yesterday ($t-1$) given all observations about umbrellas up until today ($t=2$)?

$$P\left(R^{(1)} \middle| u^{(1)}, u^{(2)}\right)$$

  - Also called *smoothing*, which refers mostly to what you do computationally, smoothing your query answers with new information; hindsight refers more directly to the query



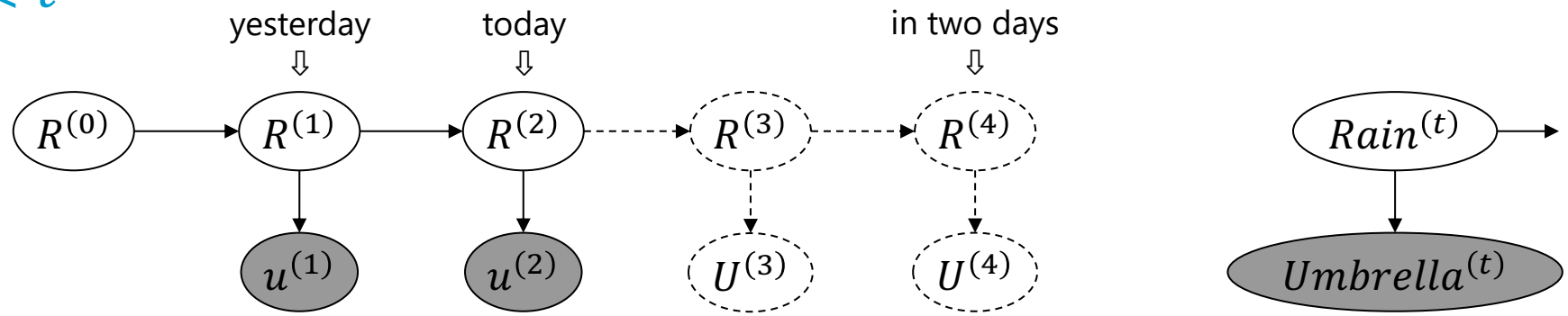


Query Answering Problem & Query Types

- Formally, query on a dynamic model M over template random variables $\boldsymbol{R}$:

$$P\left(\boldsymbol{S}^{(\pi)} \middle| \boldsymbol{e}^{(0:t)}\right)$$

  - where t the current step, $\boldsymbol{S} \subseteq \boldsymbol{R}$ query terms, $\boldsymbol{E} \subseteq \boldsymbol{R}$ evidence terms, and usually $\boldsymbol{S} \cap \boldsymbol{E} = \emptyset$
  - Filtering: $\pi = t$
  - Prediction: $\pi > t$
  - Hindsight: $\pi < t$



Interim Summary

- Modelling of sequential data
  - Assumptions: Markov-1, stationary process
  - Dynamic models consists of two episodic models
    - One to describe the initial step
    - One to describe the behaviour within a step and the transition from one step to the next
      - Copy pattern
  - Semantics: Unroll model for T steps, yielding an episodic model with known semantics (grounding, building full joint)
- Inference task: query answering
  - Query types: Filtering, prediction, hindsight

Outline: 6. Lifted Sequential Models & Inference

A. *Lifted modelling of sequences*
   - Templates, modelling
   - Semantics, inference tasks

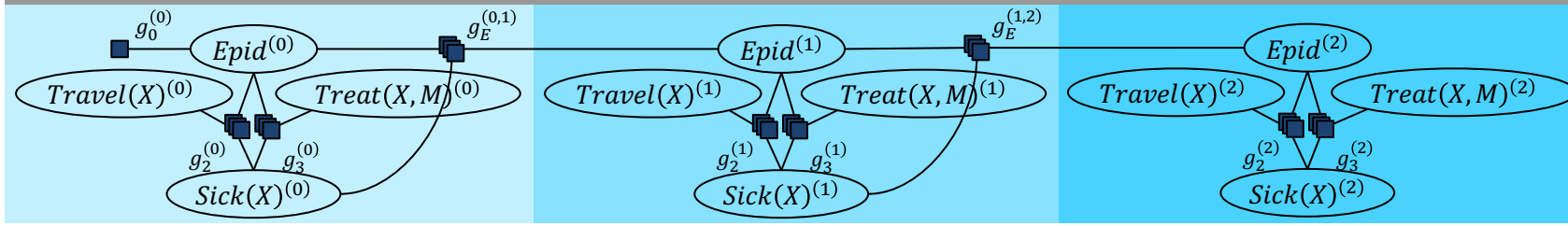
B. *Lifted dynamic inference*
   - Interfaces, lifted dynamic junction tree algorithm (LDJT)
   - Theoretical analysis: complexity
   - Keeping inference polynomial: Temporal approximate merging (TAMe)

Naïve Inference Using Unrolling

- Given a dynamic model $M = (M^0, M^{\rightarrow})$ and a query $P\left(\boldsymbol{S}^{(\pi)} \middle| \boldsymbol{e}^{(1:t)}\right)$
  1. Unroll model for $T = \max\{t, \pi\}$ steps → episodic model
  2. Answer $P\left(\boldsymbol{R}^{(\pi)} \middle| \boldsymbol{e}^{(1:t)}\right)$ with any inference algorithm (e.g., VE, LVE)
- Problems
  - Unrolled model very large → not memory-efficient
  - Unroll anew or adapt whenever T or $\boldsymbol{e}^{(1:t)}$ changes
    - Time progresses: $T{+}{+}$
      - Extend model with $M^{\rightarrow}$ for $\tau = t + 1$
      - Handle evidence extended by $\boldsymbol{e}^{(t+1)}$
    - Costly with queries of varying π per t

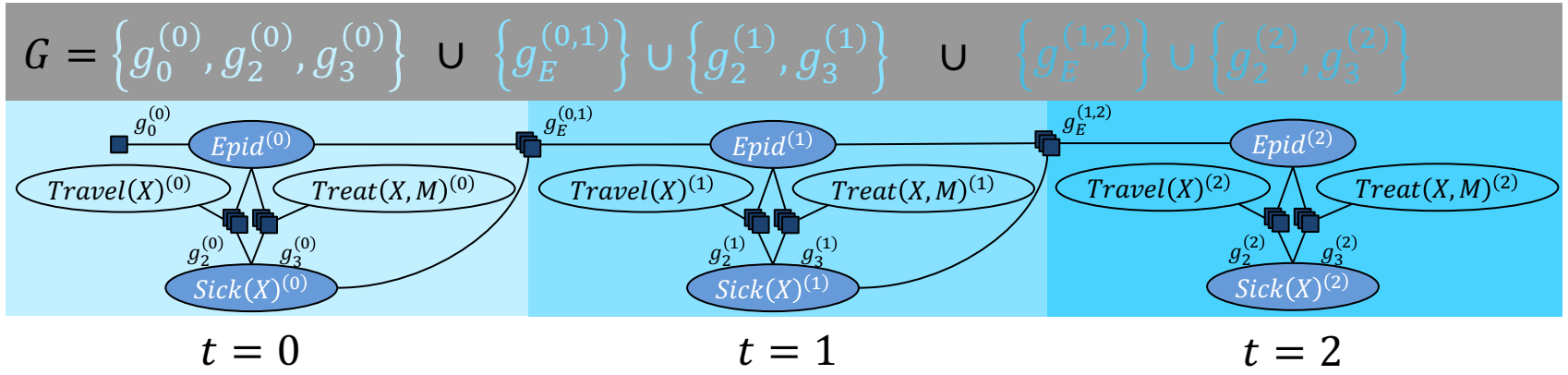
Goal: Only work with a current model $M^{(t)}$, which can handle multiple queries and progressing in time efficiently

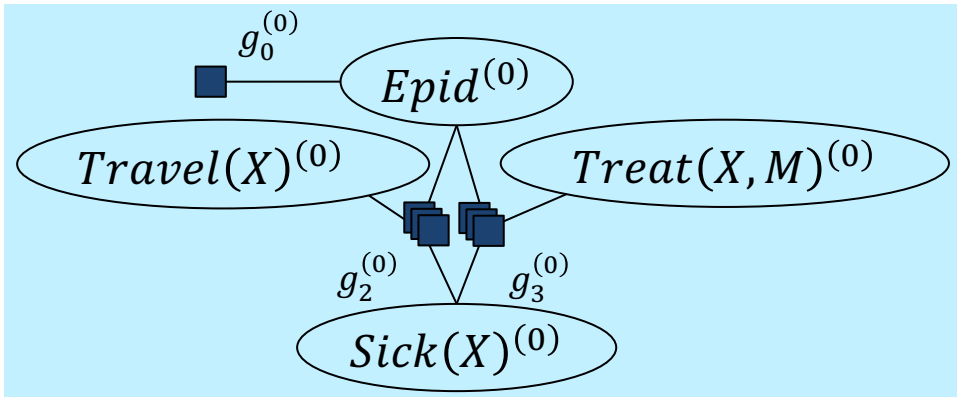
$$G = \left\{g_0^{(0)}, g_2^{(0)}, g_3^{(0)}\right\} \cup \left\{g_E^{(0,1)}\right\} \cup \left\{g_2^{(1)}, g_3^{(1)}\right\} \cup \left\{g_E^{(1,2)}\right\} \cup \left\{g_2^{(2)}, g_3^{(2)}\right\}$$

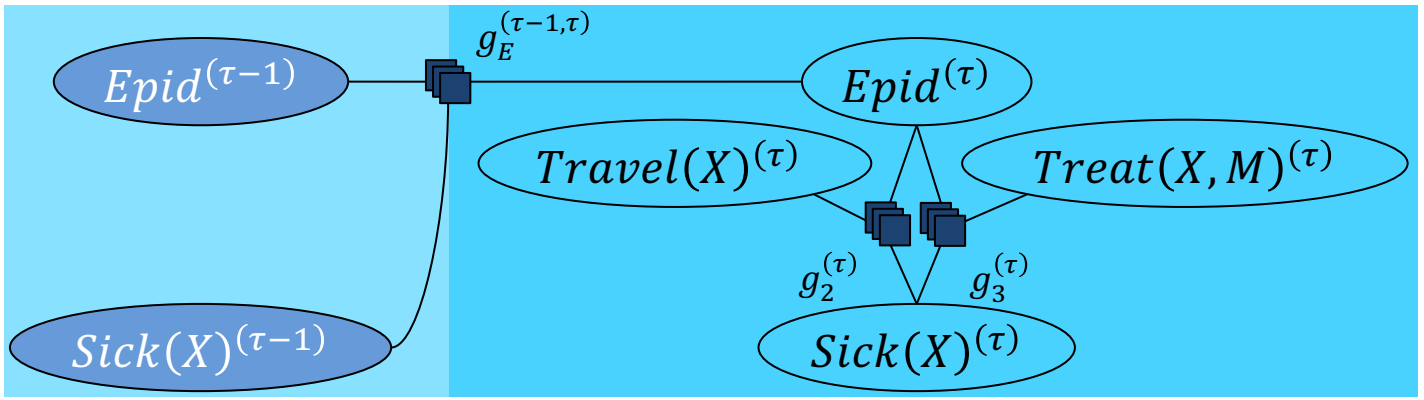


Independences in Dynamic Models

- *Observation*:
  All paths from one step to the next go through the PRVs $A^{(\tau-1)}$ of the inter-slice parfactors $g^{(\tau-1,\tau)}$ in $M^{\rightarrow}$



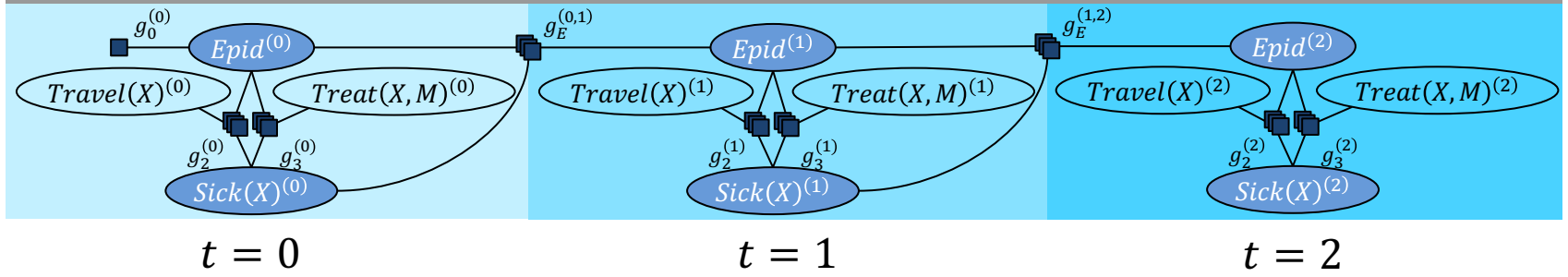




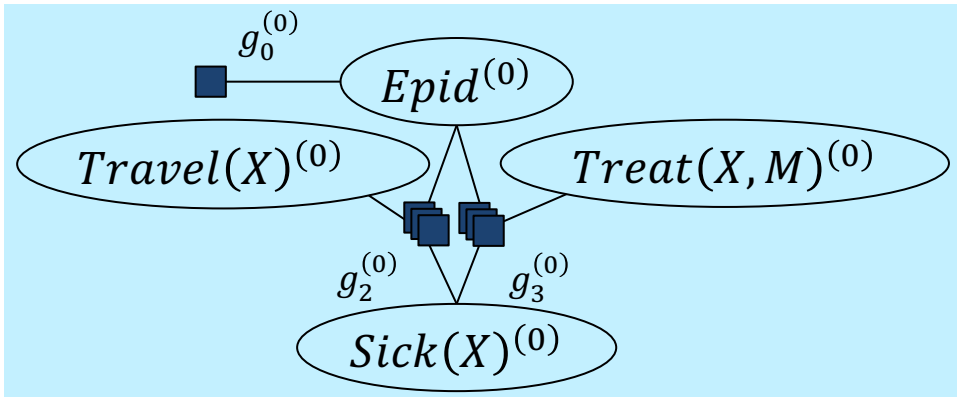
Independences in Dynamic Models

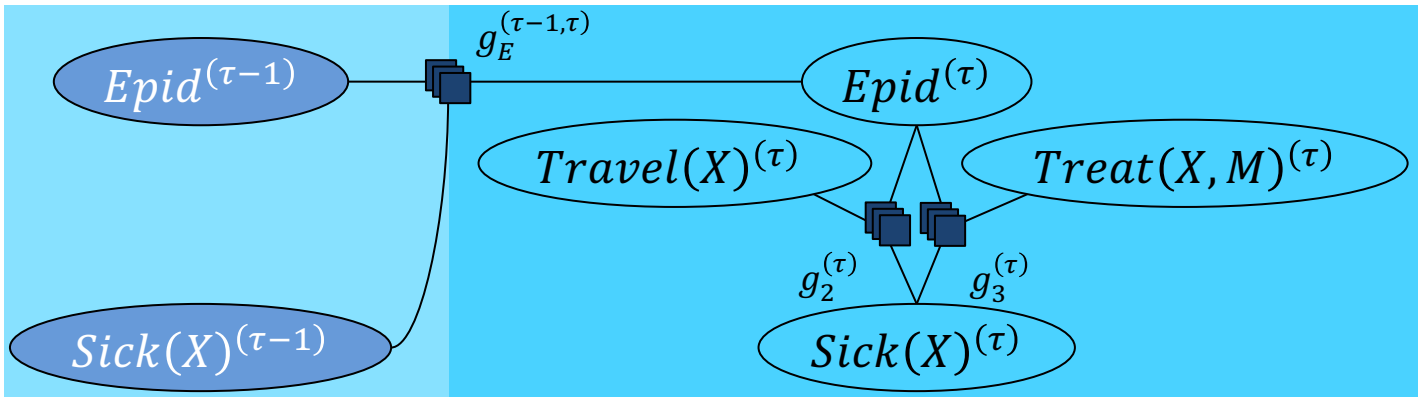
- Fact: Conditioning on these PRVs makes a current step independent of the previous steps
  - Collect information on these PRVs as a message before progressing in time

$$G = \left\{g_0^{(0)}, g_2^{(0)}, g_3^{(0)}\right\} \cup \left\{g_E^{(0,1)}\right\} \cup \left\{g_2^{(1)}, g_3^{(1)}\right\} \cup \left\{g_E^{(1,2)}\right\} \cup \left\{g_2^{(2)}, g_3^{(2)}\right\}$$



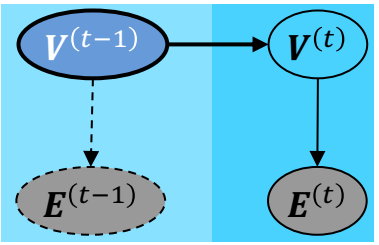
$t = 0$ $t = 1$ $t = 2$

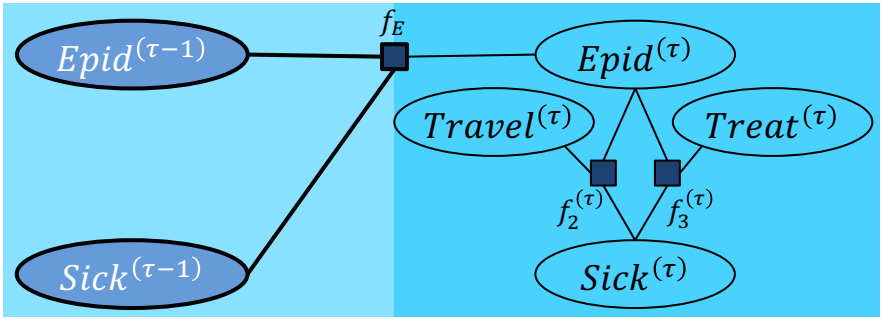




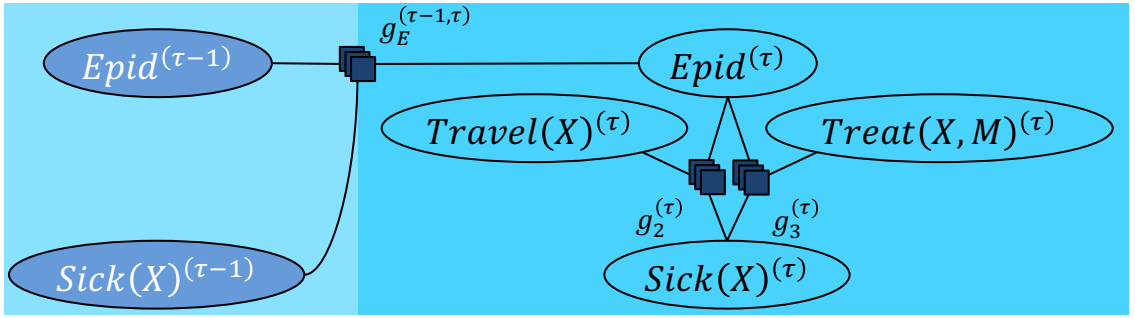
Interfaces

- Forward interface $\boldsymbol{I}^{(\tau-1)}$:
  Separating subset between $M^{(\tau-1)}$ and $M^{(\tau)}$
  - Trivial separators, usually not minimal
    - Maximal interface: $\boldsymbol{R}$
    - If $\boldsymbol{R} = \boldsymbol{V} \cup \boldsymbol{E}$: maximal interface $\boldsymbol{V}$
  - Set of PRVs with index $\tau - 1$ in $M^{\rightarrow}$
    - $\boldsymbol{I}^{(\tau-1)} = \left\{ A^{(\tau-1)} \middle| A^{(\tau-1)} \in \mathrm{rv}(M^{\rightarrow}) \right\}$
      - Can only appear in inter-slice parfactors or MLN formulas



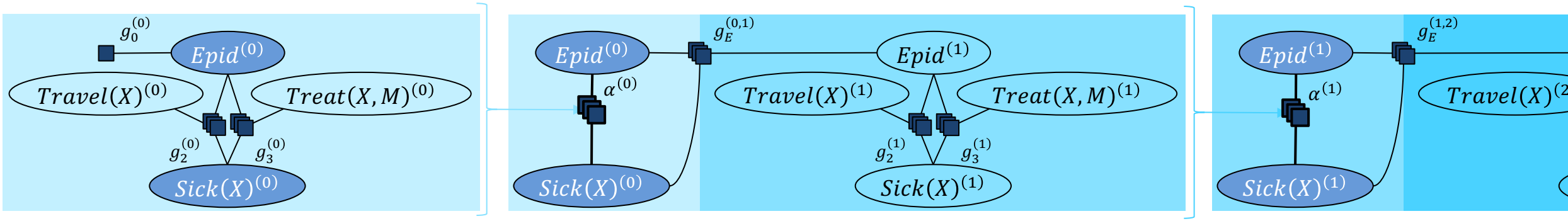


$\boldsymbol{I}$ also separates backwards from $M^{(\tau)}$ to $M^{(\tau-1)}$; there is also an explicit *backward interface*, which is determined in direction of τ to $\tau - 1$
[More details in Kevin Murphy's dissertation, 2002]



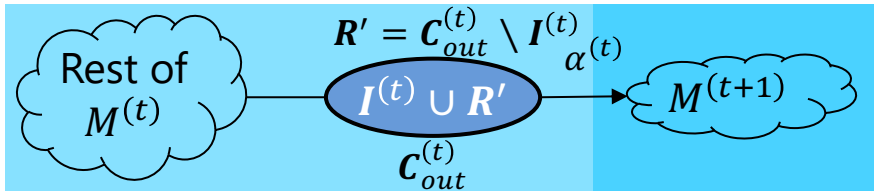
Interfaces To Move Through Time

- When inference for step t done
  - Query for $\boldsymbol{I}^{(t)}$ in current model $M^{(t)}$
    - Comparable to messages in (FO) jtrees: Calculate message $\alpha^{(t)}$ over separator $\boldsymbol{I}^{(t)}$
  - Add query result to model $M^{(t+1)}$
    - $M^{(t+1)} = M^{\rightarrow|\tau=t+1} \cup \alpha^{(t)}$
      - $M^{(t+1)}$ without connection to $M^{(0:t)}$, as all information of $M^{(0:t)}$ encoded in $\alpha^{(t)}$
- Use (FO) jtrees and (L)JT for moving through time



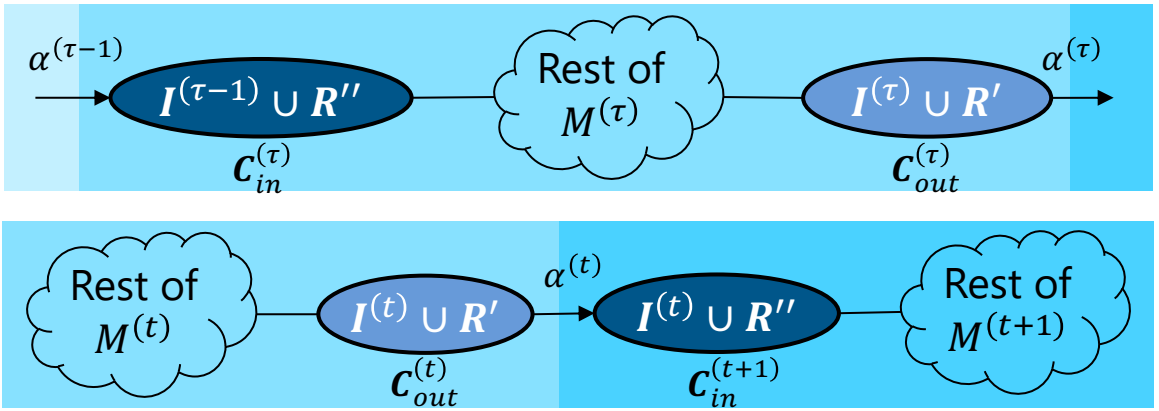
FO Jtrees and LJT To Move Through Time

- Also enables efficient multi-query answering within each slice
- Adaptations of FO jtree for dynamics: $\boldsymbol{I}^{(t)}$ has to appear in one parcluster
  - Avoids repeatedly asking large subgraph query for forward message $\alpha^{(t)}$
  - In parfactor-based model: Add parfactor $\phi_{out}(\boldsymbol{I}^{(\tau)})$ to model before FO jtree construction
    - Potentials in ϕ irrelevant for construction, could be left unspecified
    - Remove ϕ after construction
      - If not removed: Choose uniform potentials for ϕ to avoid distorting the distribution
  - Ensures that a parcluster $\boldsymbol{C}_{out}$ contains $\boldsymbol{I}^{(\tau)}$ (jtree property 2), i.e., $\boldsymbol{I}^{(\tau)} \subseteq \boldsymbol{C}_{out}$
    - $\boldsymbol{C}_{out}$ separates the slices from each other
    - So called out-cluster, as $\alpha^{(t)}$ calculated in $\boldsymbol{C}_{out}$ over $\boldsymbol{I}^{(\tau)}$ and sent forward (*outgoing*)



FO Jtrees and LJT To Move Through Time

- As $\alpha^{(t)}$ over $\boldsymbol{I}^{(t)}$ has to be added to a parcluster in the FO jtree for $t+1$, there needs to be a parcluster $\boldsymbol{C}_{in}$, which contains the interface to the previous slice
  - So called in-cluster, as forward message $\alpha^{(t)}$ over $\boldsymbol{I}^{(t)}$ is sent to $\boldsymbol{C}_{in}$ (*incoming*)
  - In parfactor-based model: Add parfactor $\phi_{in}(\boldsymbol{I}^{(\tau-1)})$ to model before FO jtree construction
    - Again, it holds that potentials in ϕ irrelevant for construction, could be left unspecified
    - Either remove ϕ after construction or choose uniform potentials for ϕ
  - Ensures that a parcluster $\boldsymbol{C}_{in}$ contains $\boldsymbol{I}^{(\tau-1)}$ (jtree property 2), i.e., $\boldsymbol{I}^{(\tau-1)} \subseteq \boldsymbol{C}_{in}$
  - In each $t > 0$ slice, in- and out-cluster



In-cluster & Out-cluster: Comparison

- Parcluster, which contains $\boldsymbol{I}^{(\tau-1)}$ → in-cluster $\boldsymbol{C}_{in}^{(\tau)}$
  - From the perspective of τ: separates past from present
  - Receives incoming information from out-cluster of preceding slice
- Parcluster, which contains $\boldsymbol{I}^{(\tau)}$ → out-cluster $\boldsymbol{C}_{out}^{(\tau)}$
  - From the perspective of τ: separates present from future
  - Sends information out towards in-cluster of next slice
  - Present becomes past in the next slice

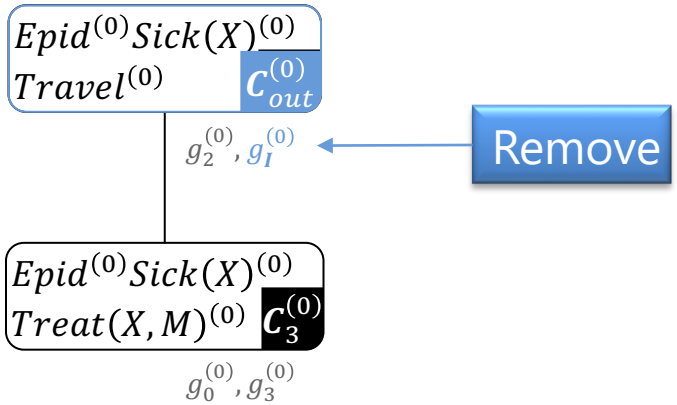
Dynamic FO Jtrees

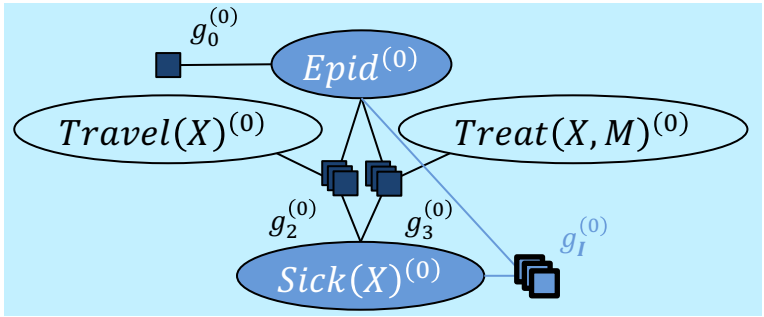
- Given an interface $\boldsymbol{I}^{(\tau)}$ for a dynamic model $M = (M^0, M^{\rightarrow})$
- Construct two FO jtrees $(J^0, J^{\rightarrow})$ for M
  - J^0 for $M^0 \cup \{\phi(\boldsymbol{I}^{(0)})\}$
    - $\boldsymbol{I}^{(0)} = \boldsymbol{I}^{(\tau)|\tau=0}$ in one parcluster (out-cluster)
  - $J^{\rightarrow}$ for $M^{\rightarrow} \cup \{\phi(\boldsymbol{I}^{(\tau-1)}), \phi(\boldsymbol{I}^{(\tau)})\}$
    - $\boldsymbol{I}^{(\tau-1)} = \boldsymbol{I}^{(\tau)|\tau=\tau-1}$ in one parcluster (in-cluster)
    - $\boldsymbol{I}^{(\tau)}$ in one parcluster (out-cluster)
    - Template FO jtree that gets instantiated by replacing τ with a step t
  - Use any approach to constructing an FO jtree
    - From lecture: Construct FO dtree, calculate clusters, transform clusters into FO jtree, minimise

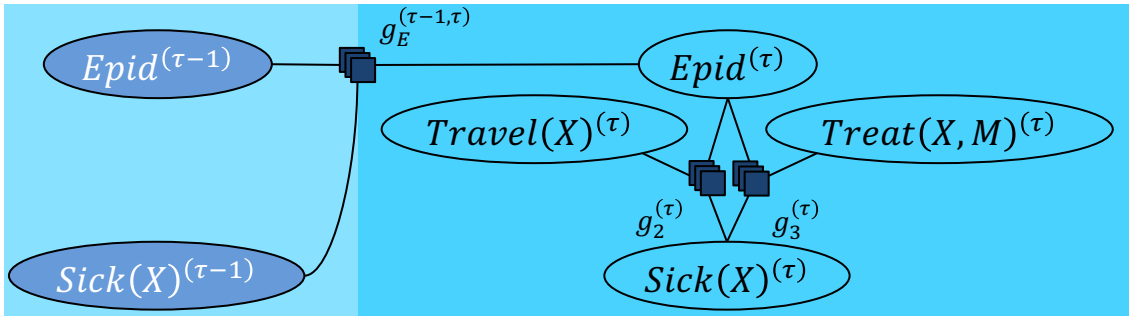
FO Jtree $J = (V, E)$ an undirected, acyclic graph with V a set of parclusters and E a set of edges for a model M.
J has to fulfil three properties:
1. Parclusters consist of PRVs of M.
2. All arguments of each parfactor occur together in a parcluster.
3. *Running intersection property.*

Dynamic FO Jtrees: Example

- Dynamic model $G = (G^0, G^{\rightarrow})$ depicted below
  - Interface $\boldsymbol{I}^{(\tau)} = \{Epid^{(\tau)}, Sick(X)^{(\tau)}\}$
- J^0 for $G^0 \cup \{g_I^{(0)}\}$
  - $g_I^{(0)} = \phi(E^{(0)}, S^{(0)})$
  - Both parclusters contain $\boldsymbol{I}^{(0)}$
    - Usually choose the parcluster, which contains $g_I^{(0)}$: $\boldsymbol{C}_2^{(0)} = \boldsymbol{C}_{out}^{(0)}$



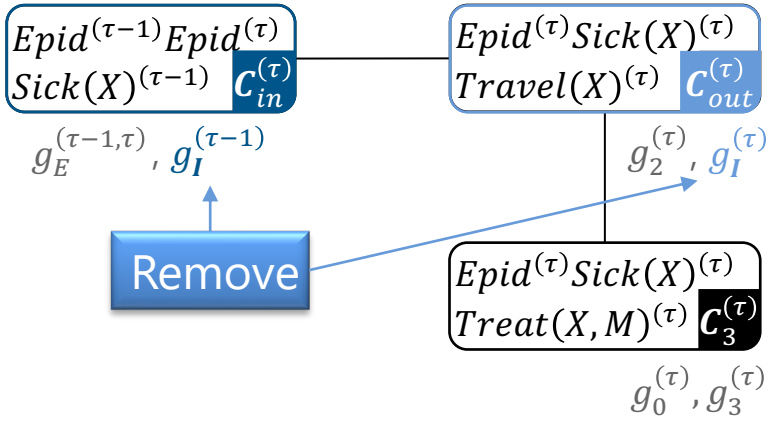


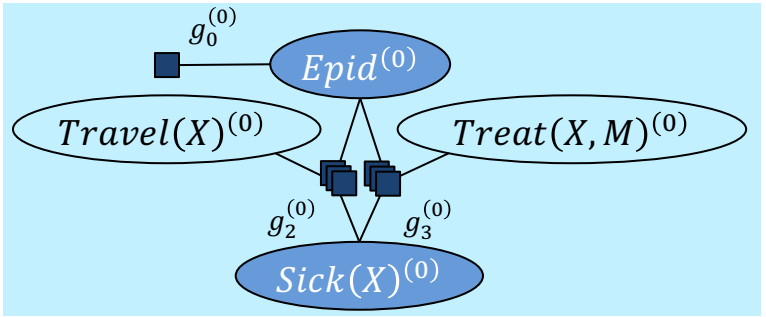


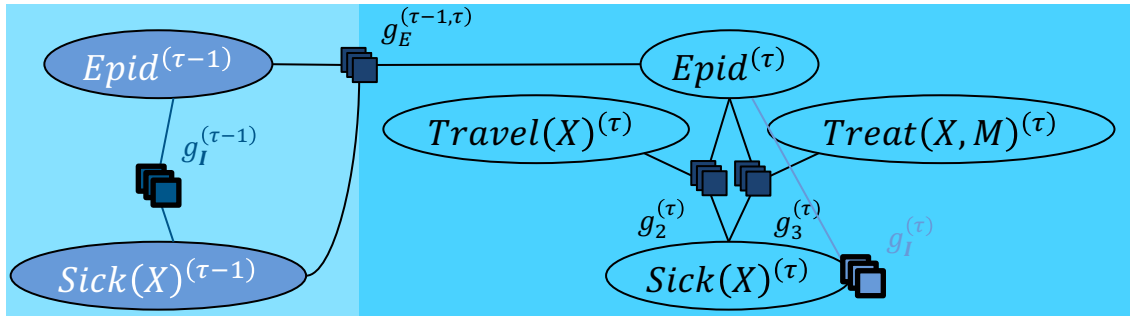
$g_I^{(0)}$ not necessary here as $\boldsymbol{I}^{(0)}$ already contained in parfactors

Dynamic FO Jtrees: Example

- Dynamic model $G = (G^0, G^{\rightarrow})$ depicted below
  - Interface $\boldsymbol{I}^{(\tau)} = \{Epid^{(\tau)}, Sick(X)^{(\tau)}\}$
- $J^{\rightarrow}$ für $G^{\rightarrow} \cup \{g_I^{(\tau-1)}, g_I^{(\tau)}\}$
  - $g_I^{(\tau-1)} = \phi(E^{(\tau-1)}, S^{(\tau-1)})$
  - $g_I^{(\tau)} = \phi(E^{(\tau)}, S^{(\tau)})$
  - In-cluster: $\boldsymbol{C}_1^{(\tau)} = \boldsymbol{C}_{in}^{(\tau)}$
  - Out-cluster: $\boldsymbol{C}_2^{(\tau)} = \boldsymbol{C}_{out}^{(\tau)}$







$g_I^{(\tau)}, g_I^{(\tau-1)}$ not necessary here as $\boldsymbol{I}^{(\tau)}, \boldsymbol{I}^{(\tau-1)}$ already contained in parfactors

Dynamic FO Jtrees: Unrolling

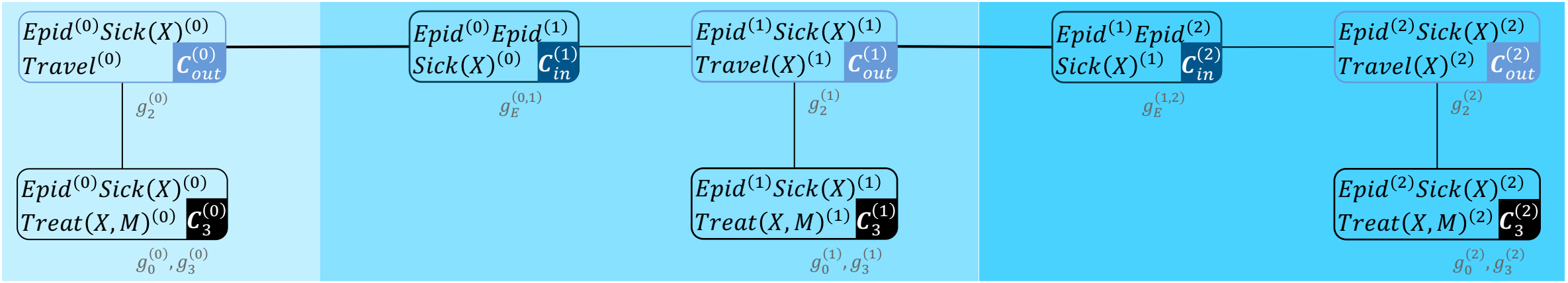
- Given two FO Jtrees $(J^0, J^{\rightarrow})$ for a dynamic model $M = (M^0, M^{\rightarrow})$ and end point T, unroll $(J^0, J^{\rightarrow})$ to form an FO jtree $J^{(0:T)}$ by
  - Instantiating FO jtrees $J^{(0)}, J^{(1)}, \ldots, J^{(T)}$, where
    - $J^{(0)} = \left(V^{(0)}, E^{(0)}\right) = J^0$
    - $J^{(t)} = \left(V^{(t)}, E^{(t)}\right) = J^{\rightarrow|\tau=t}$ for all $J^{(t)}$, $t > 0$
  - Adding an edge between the out-cluster of $J^{(t-1)}$ and the in-cluster of $J^{(t)}$ for all $t-1, t$, $t > 0$
  - Formally, $J^{(0:T)} = (V, E)$ with

$$V = V^{(0)} \cup \bigcup_{t=1}^{T} V^{(t)}$$

$$E = E^{(0)} \cup \bigcup_{t=1}^{T} E^{(t)} \cup \bigcup_{t=1}^{T} \left\{ \left\{ \boldsymbol{C}_{out}^{(t-1)}, \boldsymbol{C}_{in}^{(t)} \right\} \middle| \boldsymbol{C}_{out}^{(t-1)} \wedge \boldsymbol{C}_{in}^{(t)} \right\}$$

Dynamic FO Jtrees: Unrolling – Example

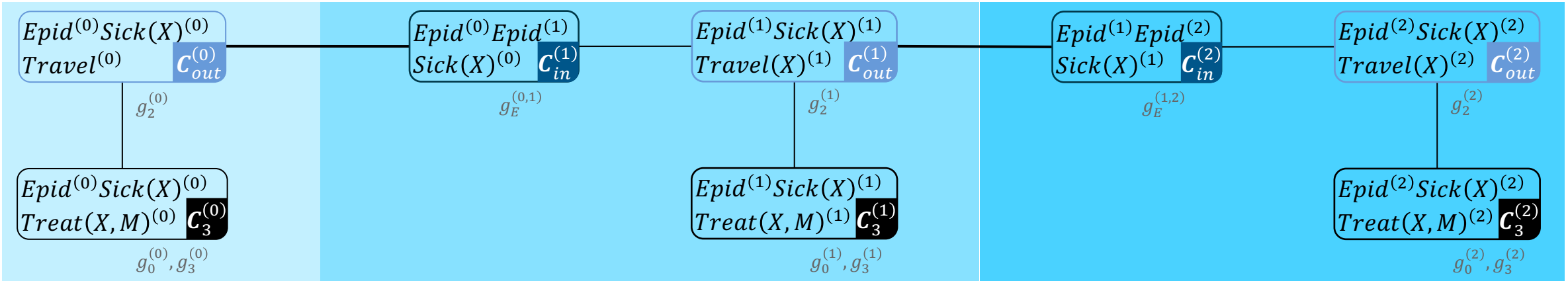
- $T = 2$



- Can use LJT for query answering in unrolled FO jtree
  - Enter evidence, pass messages, answer queries
- Problems (similar to unrolling the model)
  - Unrolled FO jtree very large → not memory-efficient
  - Unroll anew (or adapt) whenever $T = \max\{t, \pi\}$ or $\boldsymbol{e}^{(1:t)}$ change

Dynamic FO Jtrees: Unrolling – Example

- $T = 2$



- Unrolling for query answering within a slice unnecessary
  - Due to interfaces in the in- and out-clusters, the slices are independent after sending a message
  - Query answering within a slice using LJT
  - Transition to next slice: Calculate message in $C_{out}^{(t)}$ over $I^{(t)}$ and add to in-cluster $C_{in}^{(t+1)}$ in newly instantiated FO Jtree for $t + 1$

Progressing in Time

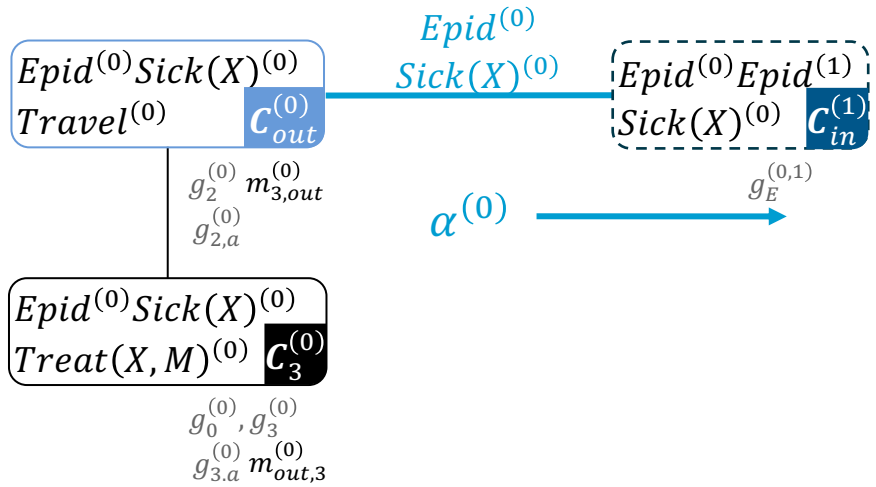
- Query answering within a slice t follows LJT (Steps 2-4 of LJT)
  - Handle evidence $\boldsymbol{e}^{(t)}$, pass messages, answer set of queries for t given all evidence up until t
  - Also ensures that all necessary information available at $\boldsymbol{C}_{out}^{(t)}$ to calculate the forward message that makes $t+1$ independent of t
- Transitioning to next slice:
  - Compute forward message $\alpha^{(t)}$ over the separator between $\boldsymbol{C}_{out}^{(t)}$ and $\boldsymbol{C}_{in}^{(t+1)}$, which includes the interface $\boldsymbol{I}^{(t)}$, based on the local model $G_{out}^{(t)}$ and the messages $m_{j,out}^{(t)}$ from neighbours within t
  - Instantiate $J^{\rightarrow}$ for $t+1$
  - Add $\alpha^{(t)}$ to the local model of $\boldsymbol{C}_{in}^{(t+1)}$
  - Drop $J^{(t)}$
  - Continue with query answering in $J^{(t+1)}$ (Steps 2-4 of LJT)

Progressing in Time: Example

- $t = 0$
  - Current FO jtree
  - Enter evidence $\boldsymbol{e}^{(0)} = \{sick(alice)^{(0)}\}$
  - Send intra-slice messages: $m_{3,out}^{(0)}$, $m_{out,3}^{(0)}$
  - Answer queries $P\left(R_i^{(0)} \middle| \boldsymbol{e}^{(0)}\right)$
- Inter-slice message: $\alpha^{(0)}$
  - Eliminate all non-separator PRVs, here $Travel(X)^{(0)}$, from the local model $G_{out}^{(0)} = \left\{g_2^{(0)}, g_{2,a}^{(0)}\right\}$ and $m_{3,out}^{(0)}$
  - Send result as message $\alpha^{(0)}$ to $\boldsymbol{C}_{in}^{(1)}$

$\alpha^{(0)}$ contains all information from $G^{(0)}$ including $\boldsymbol{e}^{(0)}$, which makes slice 0 and 1 independent.

Next: Instantiate an FO Jtree for $\tau = 1$ and add $\alpha^{(0)}$ to the local model of $\boldsymbol{C}_{in}^{(1)}$.

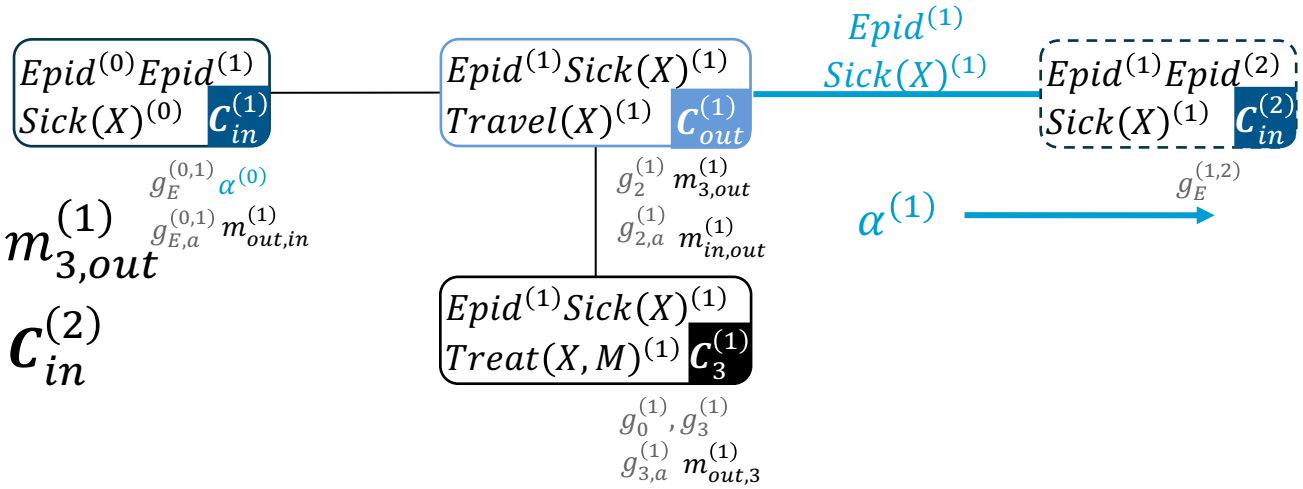


Progressing in Time: Example

- $t = 1$
  - Current FO jtree with $\alpha^{(0)}$
  - Enter evidence $\boldsymbol{e}^{(1)} = \{sick(alice)^{(1)}\}$
  - Send intra-slice messages: $m^{(1)}_{3,out}$, $m^{(1)}_{out,3}$, $m^{(1)}_{in,out}$, $m^{(1)}_{out,in}$
  - Answer queries $P\left(R^{(1)}_i \middle| \boldsymbol{e}^{(0:1)}\right)$
- Inter-slice message: $\alpha^{(1)}$
  - Eliminate $Travel(X)^{(1)}$ from $G^{(1)}_{out} = \{g^{(1)}_2, g^{(1)}_{2,a}\}$, $m^{(1)}_{in,out}$, and $m^{(1)}_{3,out}$
  - Send result as message $\alpha^{(1)}$ to $\boldsymbol{C}^{(2)}_{in}$

The information in $\alpha^{(0)}$ is distributed during the intra-slice message passing and therefore also included in $\alpha^{(1)}$. $\alpha^{(1)}$ now contains all information from $G^{(0:1)}$ including $\boldsymbol{e}^{(0:1)}$, which makes slice 1 and 2 independent.

Next: Instantiate an FO Jtree for $\tau = 2$ and add $\alpha^{(1)}$ to the local model of $\boldsymbol{C}^{(2)}_{in}$.

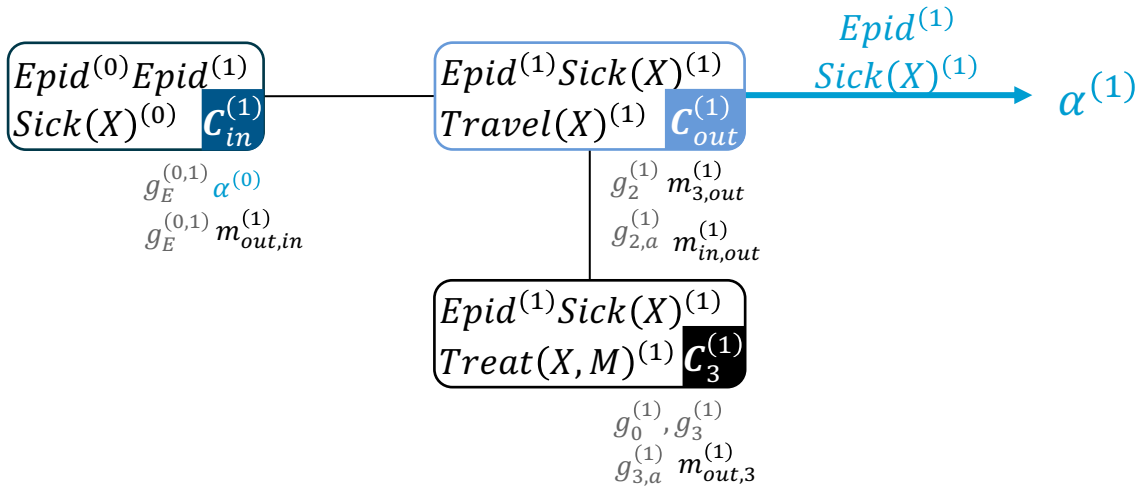


Filtering Queries

- Queries one can answer in this way: *Filtering* queries
  - $P\left(R^{(t)} \middle| \boldsymbol{e}^{(0:t)}\right)$
  - Queries for random variables / grounded instances of PRVs / parameterised queries of the current slice t given all evidence $\boldsymbol{e}^{(0:t)}$ up until t
  - Advantages
    - Only a current FO jtree necessary
    - One additional message ($\alpha^{(t)}$) to progress in time

- What about *prediction* and *hindsight* queries?
  - $P\left(R^{(\pi)} \middle| \boldsymbol{e}^{(0:t)}\right)$, $\pi \neq t$

Any ideas?



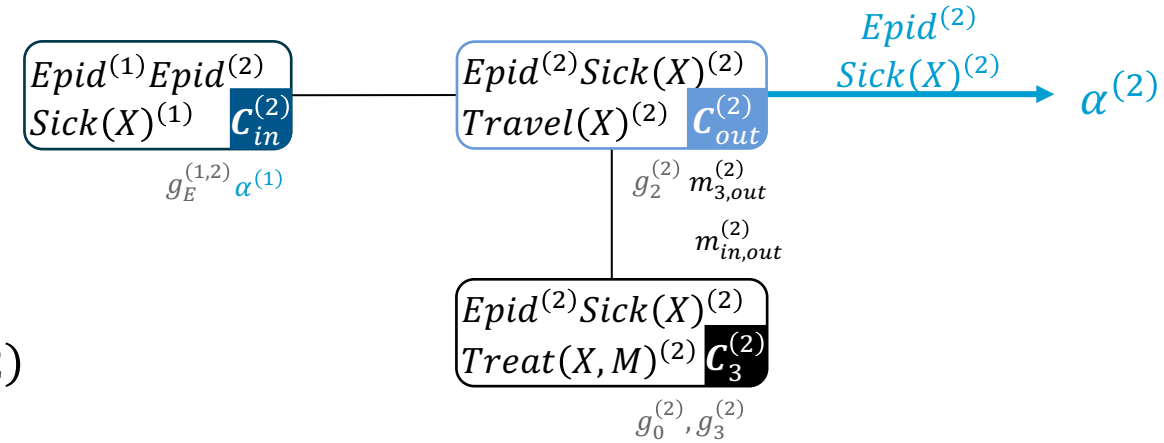
Prediction Queries

- Queries for upcoming instances of PRVs / random variables: $P\left(R^{(\pi)} \middle| \boldsymbol{e}^{(0:t)}\right), \pi > t$
  - Look into the future: Move forward in time until π without seeing evidence, i.e.,
    - Filtering $P\left(R^{(\pi)} \middle| \boldsymbol{e}^{(0:\pi)}\right)$ with empty evidence between $t+1$ and π in $\boldsymbol{e}^{(0:\pi)}$: $\boldsymbol{e}^{((t+1):\pi)} = \left\{\emptyset^{(t')}\right\}_{t'=t+1}^{\pi}$
- Query answering for prediction
  - Move forward until $t = \pi$
    - One pass from periphery to *out-cluster* as centre sufficient
  - Answer query with query term $R^{(\pi)}$
    - If only one query
      - Find parcluster $\boldsymbol{C}_i^{\pi}$, which contains $R^{(\pi)}$; do one message pass from periphery to $\boldsymbol{C}_i^{\pi}$ as centre; answer in $\boldsymbol{C}_i^{\pi}$
    - Otherwise:
      - Do a complete message pass; answer queries

Saves half the messages

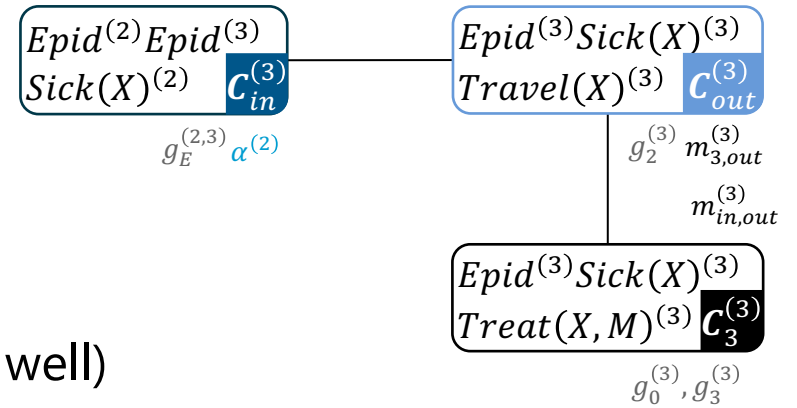
Prediction Queries: Example

- $P\left(Epid^{(3)} \middle| \boldsymbol{e}^{(0:1)}\right)$, $3 > 1$
  - No Evidence for slices 2 and 3: $\boldsymbol{e}^{(2:3)} = \{\emptyset^{(t')}\}_{t'=2}^{3}$
  - Progress in time until $t = 3$, then answer query for $Epid^{(3)}$
  - $t = 2$, current FO Jtree with $\alpha^{(1)}$ in local model of $\boldsymbol{C}_{in}^{(2)}$
    - No Evidence to enter: $\emptyset^{(2)}$
    - Send intra-slice messages towards $\boldsymbol{C}_{out}^{(2)}$ (collect all information at out-cluster)
    - Compute inter-slice message $\alpha^{(2)}$



Prediction Queries: Example

- $P\left(Epid^{(3)} \middle| \boldsymbol{e}^{(0:1)}\right)$, $3 > 1$
  - No Evidence for slices 2 and 3: $\boldsymbol{e}^{(2:3)} = \left\{\emptyset^{(t')}\right\}_{t'=2}^{3}$
  - Progress in time until $t = 3$, then answer query for $Epid^{(3)}$
  - $t = 3$, current FO Jtree with $\alpha^{(2)}$ in local model of $\boldsymbol{C}_{in}^{(3)}$
    - No Evidence to enter: $\emptyset^{(3)}$
    - If $Epid^{(3)}$ only query
      - Send intra-slice messages towards $\boldsymbol{C}_{out}^{(3)}$ (or $\boldsymbol{C}_{3}^{(3)}$)
      - Answer query in $\boldsymbol{C}_{out}^{(3)}$ (or $\boldsymbol{C}_{3}^{(3)}$)
    - Otherwise:
      - Send all messages ($m_{out,3}^{(3)}$, $m_{out,in}^{(3)}$ as well)

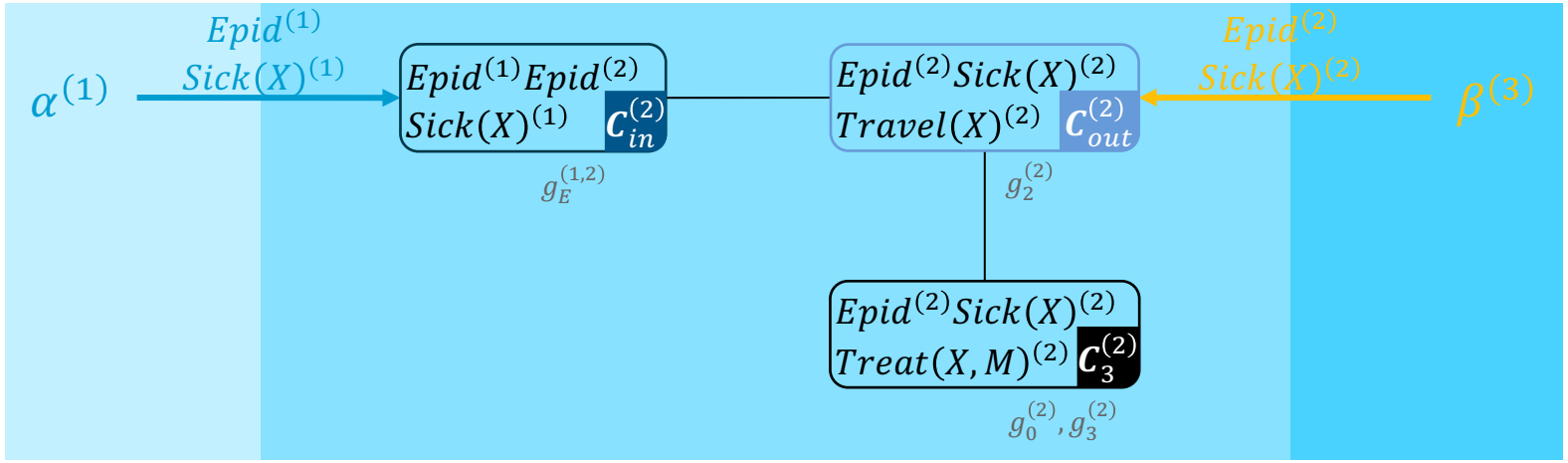


Hindsight Queries: Going Back Again

- Query for previous instances of PRVs / random variables: $P\left(R^{(\pi)} \middle| \boldsymbol{e}^{(0:t)}\right), \pi < t$
  - Looking back with the knowledge from now (t) regarding evidence, i.e.,
    - Move backwards from the current slice t and bring the information, which has been accumulated between π and t, to π; then answer a filtering query in π
- Query answering for hindsight
  - Move backwards until $t = \pi$
    - Calculate a *backward message* $\beta^{(t)}$ from $\boldsymbol{C}_{in}^{(t)}$ to $\boldsymbol{C}_{out}^{(t-1)}$ starting at t until π
      - One pass from periphery to the *in-cluster* as centre is sufficient
  - Answer query with query term $R^{(\pi)}$ in J^{π} with $\alpha^{(\pi-1)}$ in in-cluster, $\beta^{(\pi+1)}$ in out-cluster
    - Again: If only one query, one message pass from periphery to $\boldsymbol{C}_i^{\pi}$, $R^{(\pi)} \in \boldsymbol{C}_i^{\pi}$, as centre; answer in $\boldsymbol{C}_i^{\pi}$
    - Otherwise: complete message pass; answer queries

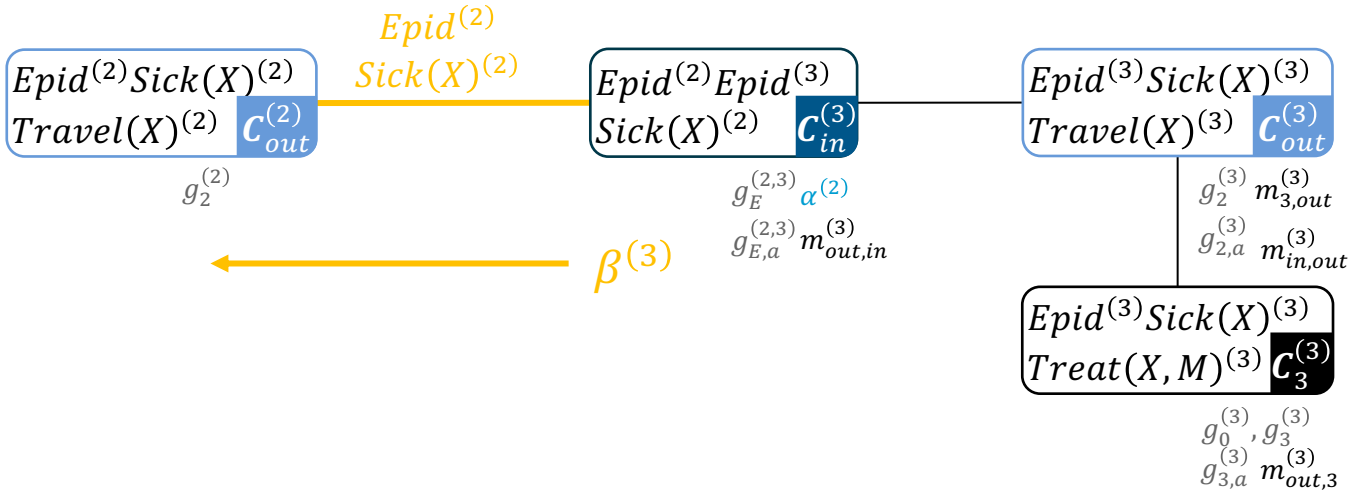
Hindsight Queries: Backward Message $\beta^{(t)}$

- Idea: Make slice independent from future (forward message backwards)
- From the perspective of slice π somewhere in the sequence from 0 to t
  - Forward message $\alpha^{(\pi-1)}$ contains all information from $M^{(0:(\pi-1))}$ including $\boldsymbol{e}^{(0:(\pi-1))}$
    - Makes $\pi - 1$ independent of π
  - Backward message $\beta^{(\pi+1)}$ contains all information from $M^{((\pi+1):t)}$ including $\boldsymbol{e}^{((\pi+1):t)}$
    - Makes π independent of $\pi + 1$
- Send from $\boldsymbol{C}_{in}^{(t)}$ to $\boldsymbol{C}_{out}^{(t-1)}$, until $t = \pi$



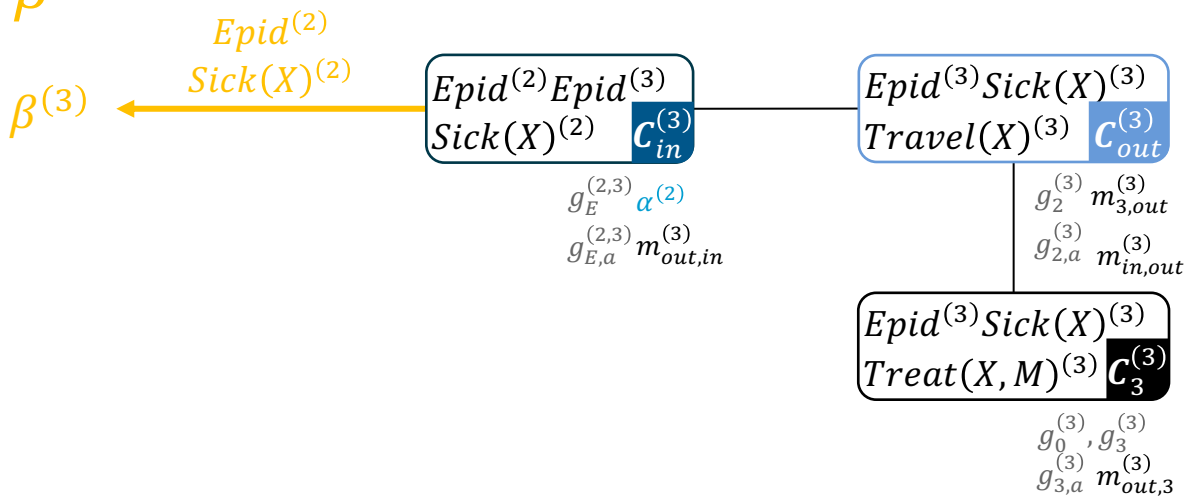
Hindsight Queries: Backward Message $\beta^{(t)}$

- Calculation
  - Eliminate all non-separator PRVs between $\boldsymbol{C}_{in}^{(t)}$ and $\boldsymbol{C}_{out}^{(t-1)}$ from the local model $G_{in}^{(t)}$ and messages $m_{j,in}^{(t)}$ from neighbours in t
    - Do not take into account forward message $\alpha^{(t-1)}$ for backward message, as $\alpha^{(t-1)}$ came from $\boldsymbol{C}_{out}^{(t-1)}$
      - Information in $\alpha^{(t-1)}$ already present at slice $t-1$
- Example:
  - Current slice $t = 3$ with evidence $\boldsymbol{e}^{(3)} = \{sick(alice)^{(3)}\}$
  - Calculate $\beta^{(3)}$ by eliminating $Epid^{(3)}$ from $g_E^{(2,3)}, g_{E,a}^{(2,3)}, m_{out,in}^{(3)}$
    - Without $\alpha^{(2)}$



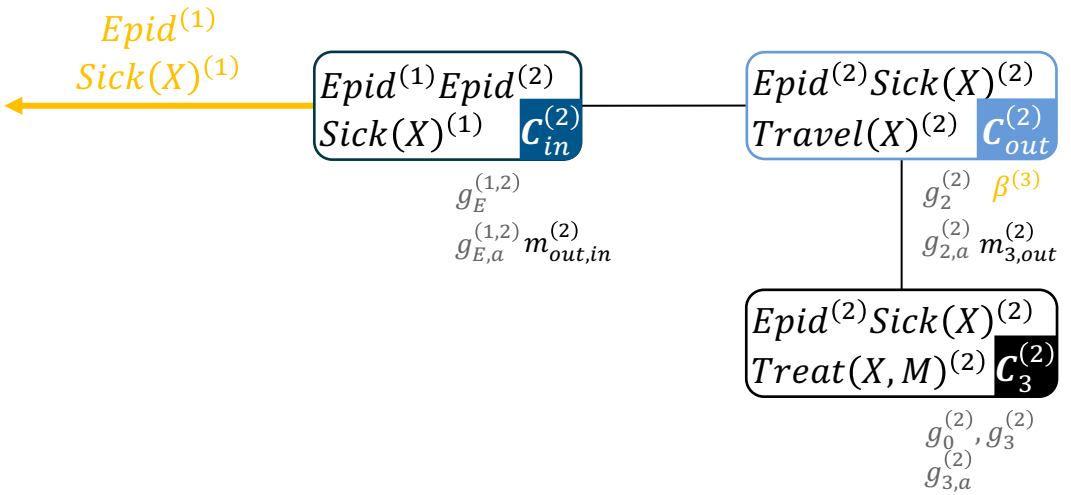
Hindsight Queries: Example

- $P\left(Epid^{(1)} \middle| \boldsymbol{e}^{(0:3)}\right)$, $1 < 3$
  - Move backwards until $t = 1$, then answer query with query term $Epid^{(1)}$
  - $t = 3$, current FO jtree with $\alpha^{(2)}$ in the local model if $\boldsymbol{C}_{in}^{(3)}$
    - Calculate backward message $\beta^{(3)}$
    - Instantiate FO jtree for $t = 2$, add $\beta^{(3)}$ to local model of $\boldsymbol{C}_{out}^{(2)}$



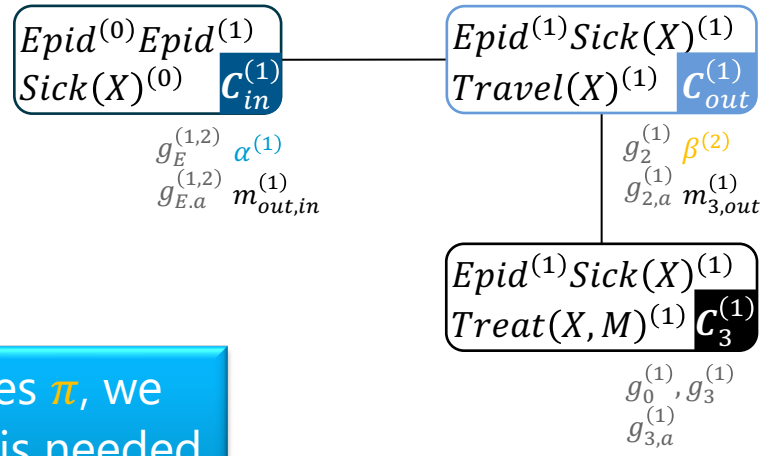
Hindsight Queries: Example

- $P\left(Epid^{(1)} \middle| \boldsymbol{e}^{(0:3)}\right)$, $1 < 3$
  - Move backwards until $t = 1$, then answer query with query term $Epid^{(1)}$
  - $t = 2$, current FO jtree with $\beta^{(3)}$ in local model of $\boldsymbol{C}_{out}^{(2)}$
    - Enter evidence $\boldsymbol{e}^{(2)} = \{sick(alice)^{(2)}\}$
    - Send messages towards $\boldsymbol{C}_{in}^{(2)}$
    - Calculate backward message $\beta^{(2)}$ $\beta^{(2)}$
    - Instantiate FO jtree for $t = 1$, add $\beta^{(2)}$ to local model of $\boldsymbol{C}_{out}^{(2)}$



Hindsight Queries: Example

- $P\left(Epid^{(1)} \middle| \boldsymbol{e}^{(0:3)}\right)$, $1 < 3$
  - Move backwards until $t = 1$, then answer query with query term $Epid^{(1)}$
  - $t = 1$, current FO jtree with $\beta^{(2)}$ in local model of $\boldsymbol{C}_{out}^{(1)}$, $\alpha^{(0)}$ in local model of $\boldsymbol{C}_{in}^{(1)}$
    - Enter evidence $\boldsymbol{e}^{(1)} = \{sick(alice)^{(1)}\}$
    - If $Epid^{(1)}$ only query
      - Send intra-slice messages towards $\boldsymbol{C}_{in}^{(1)}$ (or one of the other two)
      - Answer query in $\boldsymbol{C}_{in}^{(1)}$
    - Otherwise: Send all messages



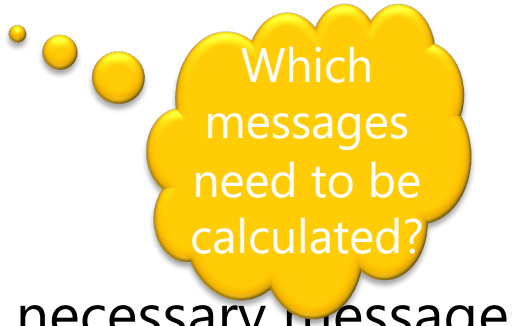
To support hindsight queries for arbitrary slices π, we have to store all forward messages, as $\alpha^{(\pi-1)}$ is needed for query answering in π, next to all evidence $\boldsymbol{e}^{(0:t)}$.

Sequential Query Answering

- Forward-Backward Algorithm for answering filtering, prediction, hindsight queries
- Given: Set of queries $\boldsymbol{Q}^{(0:T_q)} = \left\{\left\{\boldsymbol{Q}_i^{(\pi_i)}\right\}_{i=1}^{m_t}\right\}_{t=0}^{T_q}$
  - With increasing evidence $\boldsymbol{e}^{(0:t)}, t \in \{0, \dots, T_e\}$
  - Also possible to consider in online setting: one query and one evidence stream over increasing t
  - Go through $\boldsymbol{Q}^{(0:T_q)}$ with increasing t and through the queries $\left\{\boldsymbol{Q}_i^{(\pi_i)}\right\}_{i=1}^{m_t}$ given evidence $\boldsymbol{e}^{(0:t)}$ based on type and slice π_i; answer in the following order in each t:
    1. Filtering queries $\boldsymbol{Q}_i^{(\pi_i)}, \pi_i = \mathrm{t}$
    2. Prediction queries $\boldsymbol{Q}_i^{(\pi_i)}$, ordered by increasing π_i ($\pi_i > t$)
    3. Hindsight queries $\boldsymbol{Q}_i^{(\pi_i)}$, ordered by decreasing π_i ($\pi_i < t$)
       - Order of 2. and 3. can be exchanged

Sequential Query Answering: Answering Queries

- Given the current slice t, query terms $\left\{\boldsymbol{Q}_i^{(\pi_i)}\right\}_{i=1}^{m_t}$, and a completed message pass in t
  - For all $\boldsymbol{Q}_i^{(\pi_i)}$ with $t = \pi_i$, answer $\boldsymbol{Q}_i^{(\pi_i)}$ in $J^{(t)}$
  - Keep $J^{(t)}$ and move through time with t', initialised with $t' \leftarrow t$
  - For all $\boldsymbol{Q}_i^{(\pi_i)}$ with $t < \pi_i$, move forward without evidence until $t' = \max_i \pi_i$:
    - Instantiate FO jtree $J^{(t')}$ and calculate necessary messages
    - Whenever $t' = \pi_i$, answer $\boldsymbol{Q}_i^{(\pi_i)}$
  - For all $\boldsymbol{Q}_i^{(\pi_i)}$ with $t > \pi_i$, move backward until $t' = \min_i \pi_i$
    - Instantiate FO jtree $J^{(t')}$, enter evidence $\boldsymbol{e}^{(t')}$, and calculate necessary messages
    - Whenever $t' = \pi_i$, answer $\boldsymbol{Q}_i^{(\pi_i)}$



Lifted Dynamic Junction Tree Algorithm (LDJT)

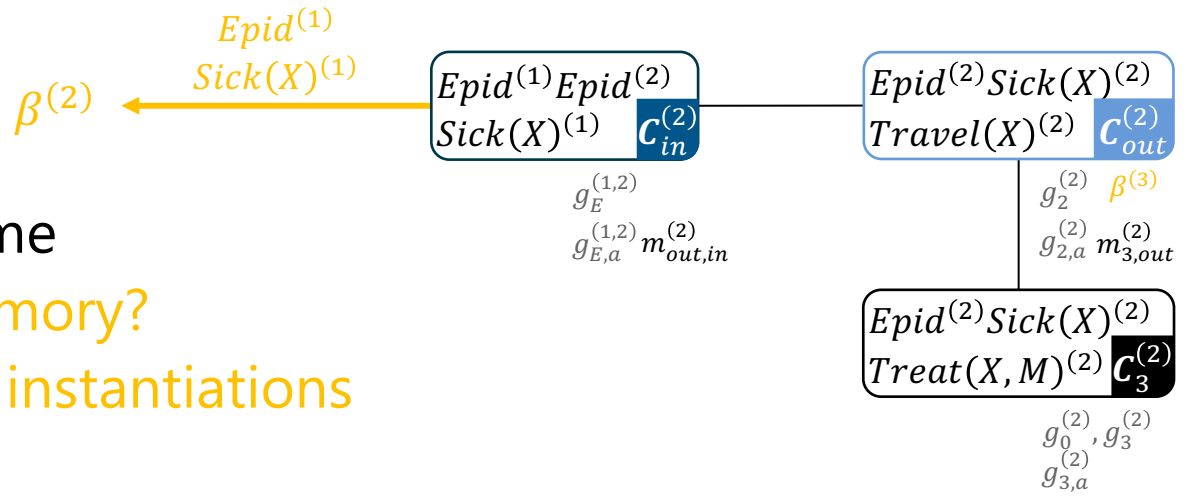
procedure LDJT($(M^0, M^{\rightarrow}), \boldsymbol{Q}^{(0:T_q)}, \boldsymbol{e}^{(0:T_e)}$)
  Construct FO jtrees $(J^0, J^{\rightarrow})$ for $M^0, M^{\rightarrow}$
  for t **in** $0 \ldots T_q$ **do**
    Instantiate $J^{(t)}$ and drop $J^{(t-1)}$ ▷ J^0 if $t = 0$; $J^{\rightarrow}$ otherwise
    Add $\alpha^{(t-1)}$ to Incluster von $J^{(t)}$ hinzu ▷ if $t > 0$
    Enter evidence $\boldsymbol{e}^{(t)}$ in $J^{(t)}$
    Send messages in $J^{(t)}$
    Answer queries $\boldsymbol{Q}^{(t)}$ ▷ Proceed as described on previous slide
    Calculate $\alpha^{(t)}$

LDJT

- In online setting, two streams of incoming queries $\boldsymbol{Q}^{(t)}$ and evidence $\boldsymbol{e}^{(t)}$
  - Instead of inputs $\boldsymbol{Q}^{(0:T_q)}, \boldsymbol{e}^{(0:T_e)}$

Approaches to Moving Backwards

- Store forward messages for arbitrary hindsight queries
  - Forward message of in-cluster needed for hindsight query in that slice
- Moving backwards means calculating messages anew in each passing slice
  - Slice t backwards without queries
    - Calculate one pass with $n-1$ messages, n number of parclusters in J_t
  - Slice t backwards with queries
    - Calculate complete message pass with $2(n-1)$ messages
  - Advantage
    - Only one FO jtree in memory at a time
- What if LDJT keeps the old FO jtrees in memory?
  - Instantiate on-demand (so far) vs. keep instantiations



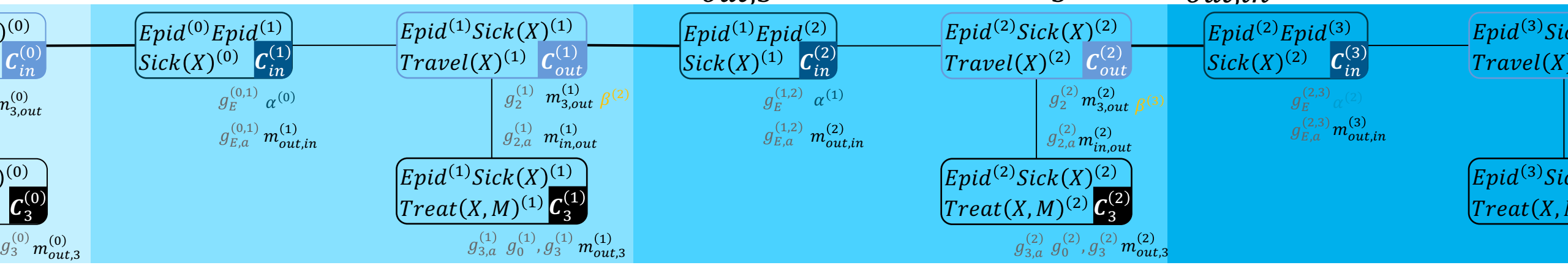
Approaches to Moving Backwards: Keeping Instantiations

- What does LDJT pay in memory?
  - Local models, intra-slice messages for each slice
- What runtime can LDJT save?
  - Entering evidence not necessary, as evidence already entered during moving forward
  - Adaptive message passing possible: Only update messages that change due to β_{t+1}
    - Slice t backwards without queries
      - β_{t+1} new information arriving at out-cluster, has to be passed to in-cluster
      - Calculate messages on path between out-cluster and in-cluster → up to $n-1$ messages, n number of parclusters in J_t (Worst Case: all parclusters in sequence with in- and out-cluster at the ends)
    - Slice t backwards with queries
      - β_{t+1} new information arriving at out-cluster, has to be passed to all other parclusters
      - Calculate one message pass from out-cluster outward to periphery with $n-1$ messages

Approaches to Moving Backwards: Example – Messages to Calculate

- In $t = 2$ (without query)
  - Keep instantiations: $m^{(2)}_{out,in}$ (on path)
  - Instantiate on-demand: $m^{(2)}_{3,out}, m^{(2)}_{out,in}$ (towards C^{in}_2)
- In $t = 1$ (with query)
  - Keep instantiations: $m^{(1)}_{out,3}$ (outwards to $\boldsymbol{C}^{(1)}_3$) + $m^{(1)}_{out,in}$ (for arbitrary

Hindsight query
$P(Treat^{(1)} | \boldsymbol{e}^{(0:3)})$



Unrolled FO jtree in memory

Approaches to Moving Backwards

	Keep instantiations	Instantiate on-demand
Slices without queries	$\leq n-1$	$n-1$
Slices with arbitrary queries	$n-1$	$2(n-1)$
Additional memory for hindsight compared to filtering/prediction	Local models, intra-slice messages, $\alpha^{(t)}$ messages	$\alpha^{(t)}$ messages

- Keeping all instantiations in memory usually not feasible
  - Additionally, goal was to build a memory-efficient algorithm that does not keep the whole model from the first slice onward in memory
- Trade off runtime vs. memory
  - Keep last k instantiations and instantiate on-demand for queries with larger lag
    - Analyse data (queries) to find out typical lag

Queries over Multiple Slices

- So far, assumed that query terms of a single query refer to the same slice π
- What about query terms from different slices?
  - Example: $P\left(Treat^{(1)}, Travel^{(2)} \middle| \boldsymbol{e}^{(0:3)}\right)$ with $t = 3 \rightarrow min = 1, max = 2$ in query
- Query answering (one of many possibilities):
  - Unroll FO jtree for those slices covered by query: $J^{(min:max)}$
    - $min =$ minimum slice in query, $max =$ maximum slice in query
    - $\alpha^{(min-1)}$ in in-cluster of $J^{(min)}$; if $max < t$, $\beta^{(max+1)}$ in out-cluster of $J^{(max)}$
  - Enter evidence $\boldsymbol{e}^{(min:max)}$
  - Find subgraph covering the query terms
  - Send messages in $J^{(min:max)}$ from periphery towards border of subgraph
  - Answer query on submodel of local models of $J^{(min:max)}$ and messages at the border

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

Fusion

- Test each message m_{ij} for each PRV A to eliminate and each separator PRV S
  - If Cond. 1 holds: continue (no groundings)
  - Else if Cond. 2 and Cond. 3 holds: continue
  - Else: mark m_{ij} (grounding); continue with next m_{ij}
- For each message m_{ij} marked:
  - Merge parclusters $\boldsymbol{C}_i, \boldsymbol{C}_j$ — Fusion

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)

Conditions on Groundings

- Problem with Cond. 1 induced using count conversions on logical variables $lv(S) \setminus lv(A)$:
  - Logical variables that were previously not counted are now counted
  - All receiving parclusters need to be able to handle counted versions, which needs to be checked
    - If newly counted logical variable arrives at 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 logical variable has to be counted in G_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.

Complexity Analysis & Runtimes

Lifted Dynamic Junction Tree Algorithm (LDJT)

Complexity

- LJT complexity for message passing and query answering

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

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

Lifted calculations — Largest parfactor size

  - 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,

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

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

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

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,

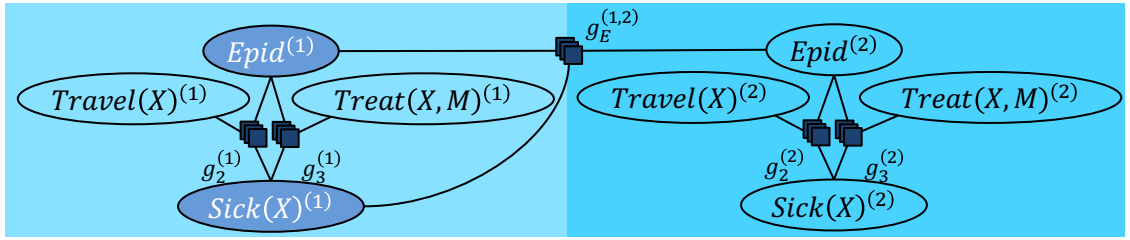
$$
\begin{aligned}
& T \cdot (O_{MP} + O_{QA}) + T^2 \cdot (O_{MP} + O_{QA}) + M \cdot O_{QA} \\
&= (T \cdot n_J + T + T^2 \cdot n_J + T^2 + M) \cdot O_{QA} \\
&= O\left(\left((T^2 + T) \cdot n_J + M\right) \cdot \log_2(n) \cdot r^{w_g} \cdot n^{r_\# w_\#}\right) \\
&= O\left(\left((T^2 + T) \cdot n_J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

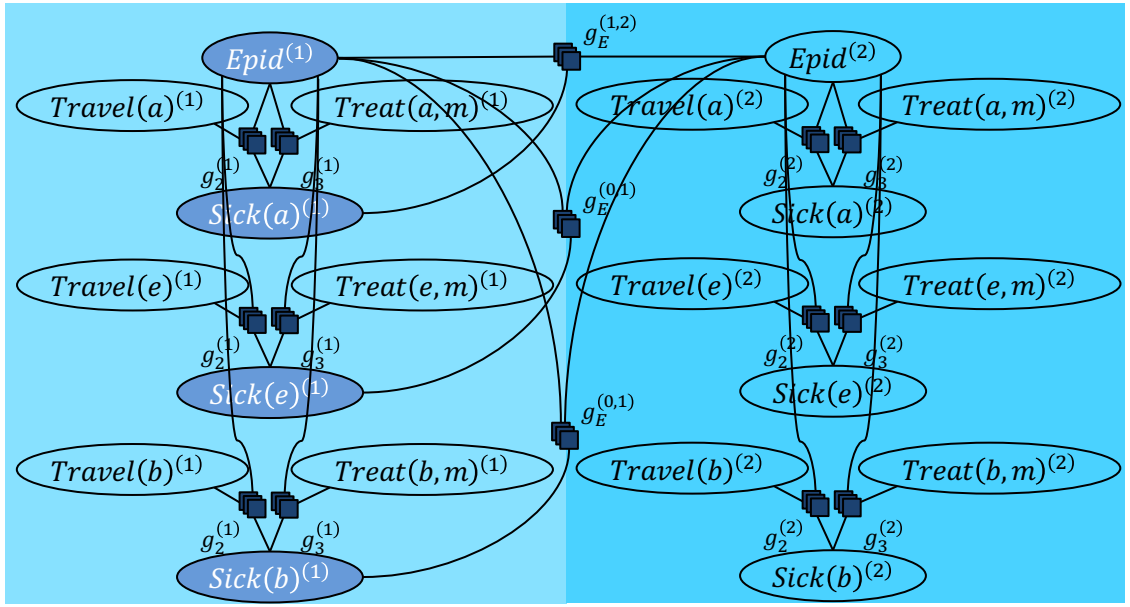
Compare filtering complexity

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

Comparison to Ground Inference

- 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, $w = w_g$, $w_\# = 0$
- In sequential case,
  - Earnings because of lifting the interface
    - Even without count conversions, $w \gg w_g$
  - Example: Grounding with
    - $\mathrm{dom}(X) = \{a, e, b\}$
    - $\mathrm{dom}(M) = \{m\}$





Runtimes

- Algorithms for comparison
  - LDJT as presented here
  - DJT: propositional interface algorithm
  - LJT FOJT: unrolling the FO jtree
  - 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

LJT FOJT performs similar to LDJT as they perform the same calculations only LJT FOJT has to keep the unrolled model in memory.

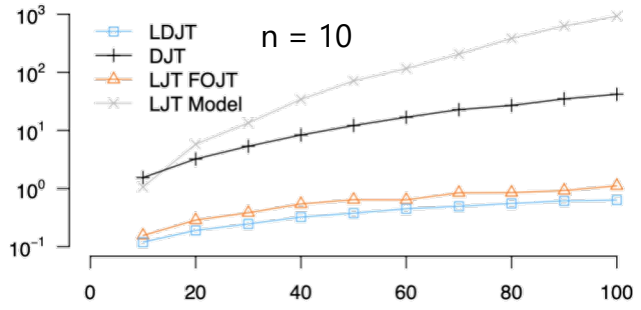


Fig. 1

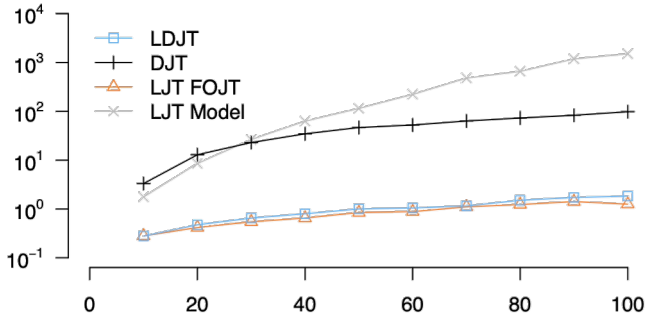


Fig. 3

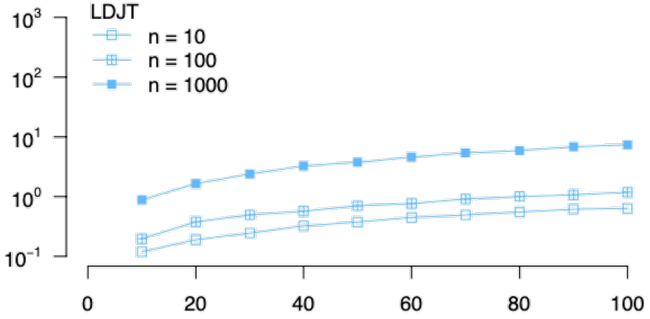


Fig. 2

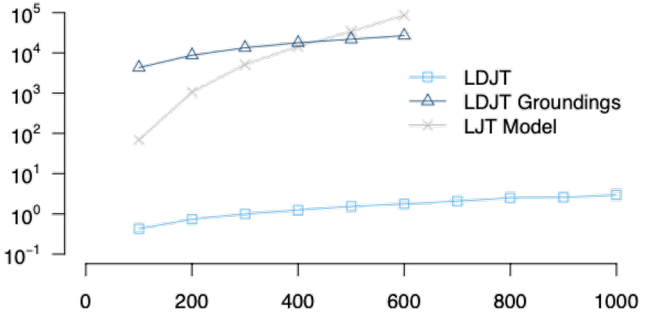
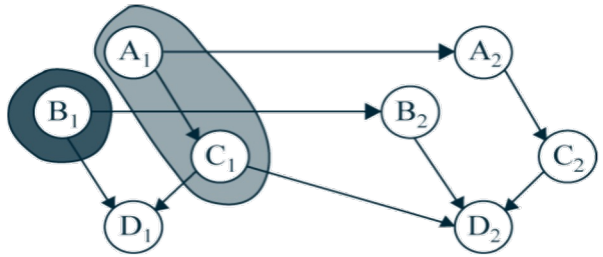


Fig. 4

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 $\{B\}$:

$$\alpha_{\tau} = \{\text{LVE}-\text{MSG}(G_{\tau}^{out}, \{A, C\}), \text{LVE}-\text{MSG}(G_{\tau}^{out}, \{B\})\}$$

  - 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

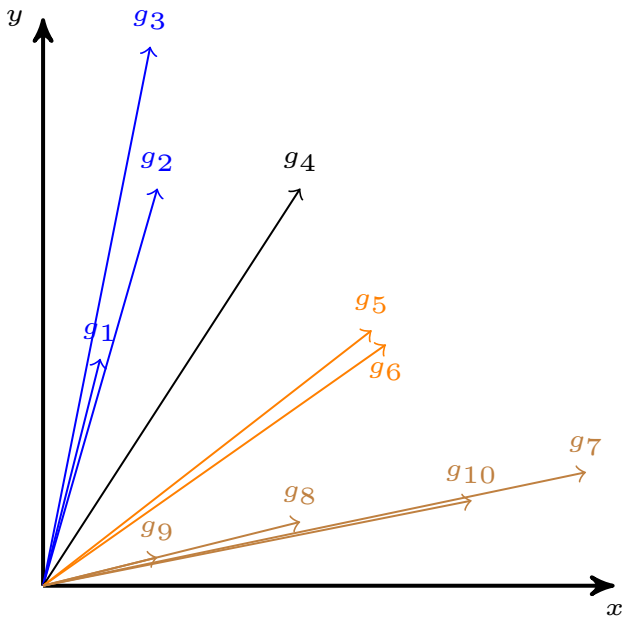
- Factored-frontier algorithm (Murphy, 2002)
  - Remember:
    - Probability propagation on polytree BNs: Send messages directly between the nodes
    - Probability propagation in general graphs called (loopy) belief propagation (BP): Send messages directly between the nodes
      - Similar to the colour 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

Temporal Approximate Merging (TAMe)



The problem of keeping inference polynomial

The Problem with Evidence

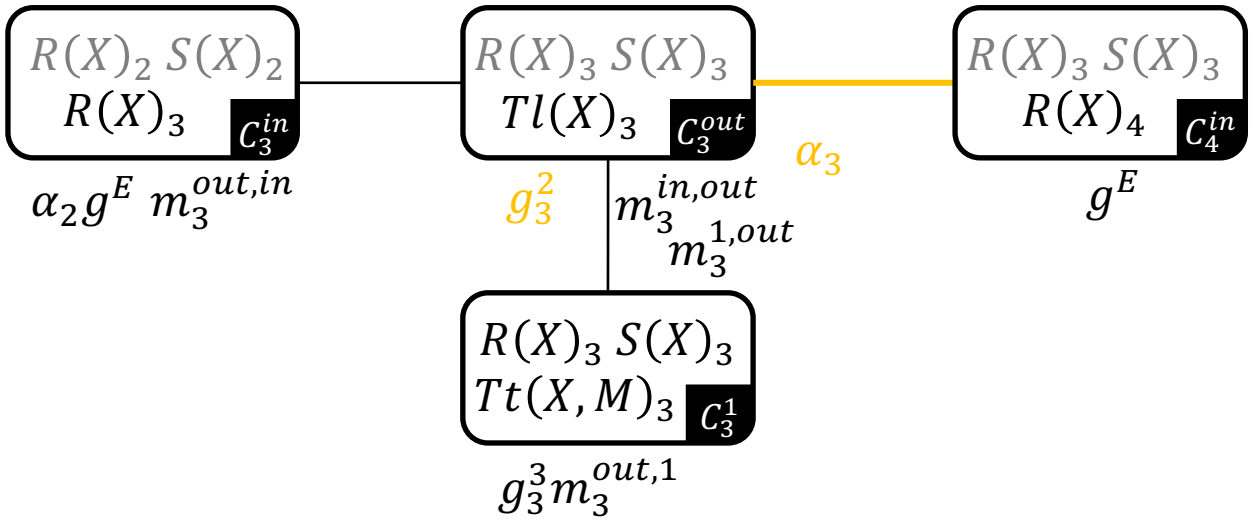
- Evidence can ground a model over time
- Non-symmetric evidence
  - Observe evidence for some instances in one time step
  - Observe evidence for a subset of these instances in another time step
  - Splits a logical variable slowly over time
- Vanilla junction trees for each time step
  - Without any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

Marcel Gehrke, Ralf Möller, and Tanya Braun. Lifted Taming Reasoning in Temporal Probabilistic Relational Models Explanation. In *ECAI-20 Proceedings of the 24th European Conference on Artificial Intelligence*, 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

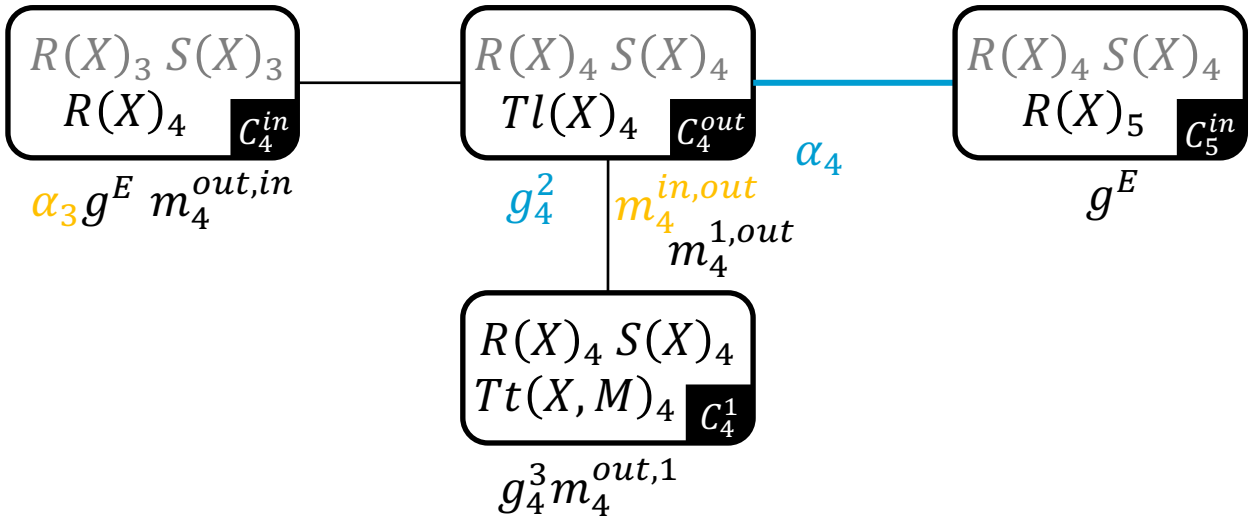


Evidence over Time

- 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 $X \neq x_2$

X is slowly grounded

- 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



Undoing Splits

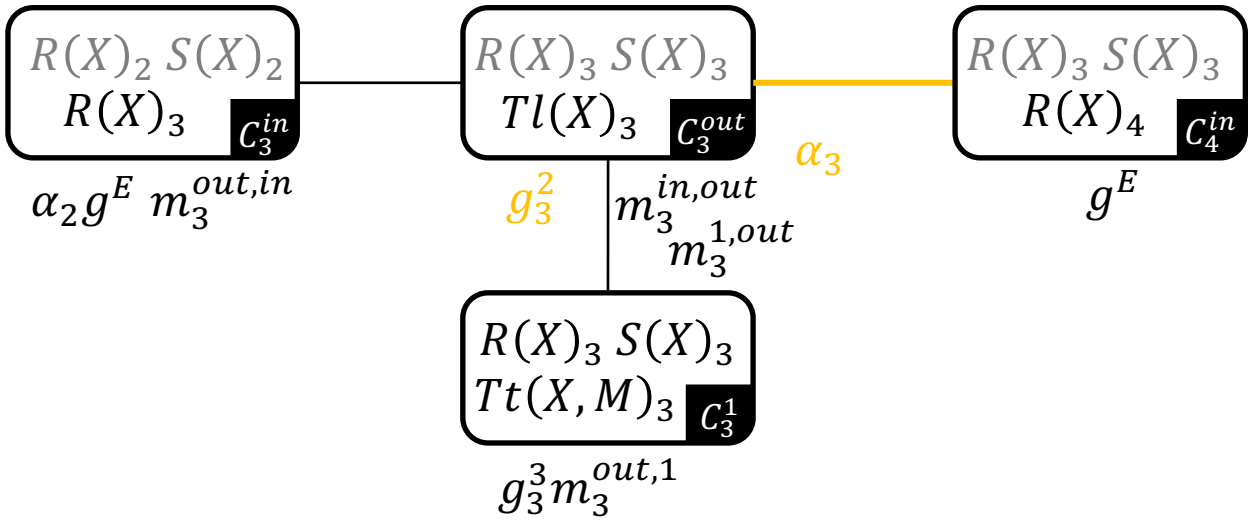
- Need to undo splits to

  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?

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



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

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.,
    - $\mathrm{dom}(X) = \{x_1\}$

$R(X)$	$S(X)$	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

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

Comparing Parfactors

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

$R(X)$	$S(X)$	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

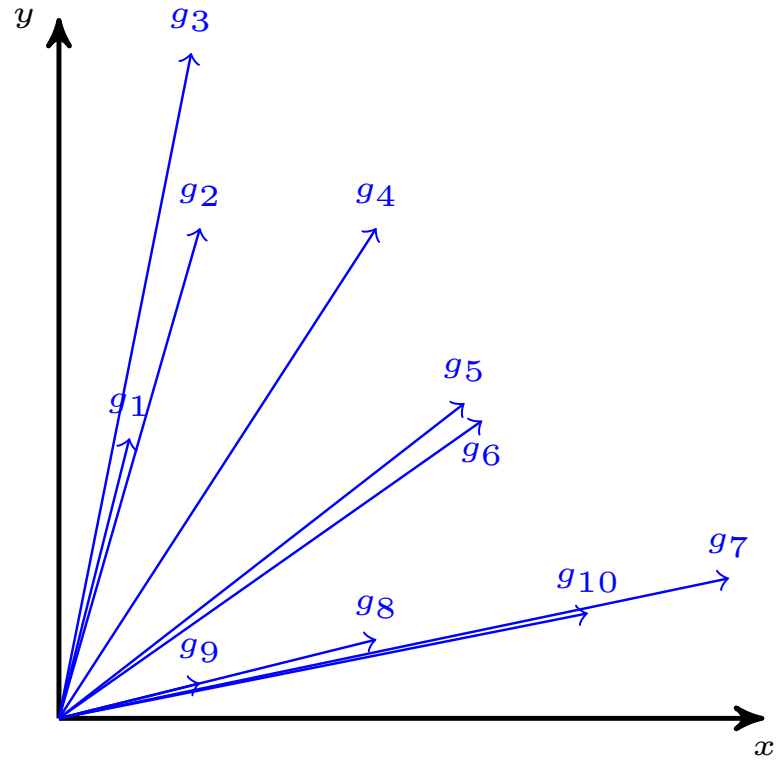
    - $P(S(x_1) = true)$: $\frac{4}{10}$ $\frac{4}{10}$
    - $P(R(x_1) = true)$: $\frac{3}{10}$ $\frac{3}{10}$
    - $P(S(x_1) = true, R(x_1) = true)$: $\frac{1}{10}$ $\frac{0.9}{10}$

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



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

  - $\cos(\theta) = \frac{0 \cdot 2 + 7 \cdot 4 + 4 \cdot 2 + 1 \cdot 4}{\sqrt{0+49+16+1} \cdot \sqrt{4+16+4+16}} \sim 0.7785$

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

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

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

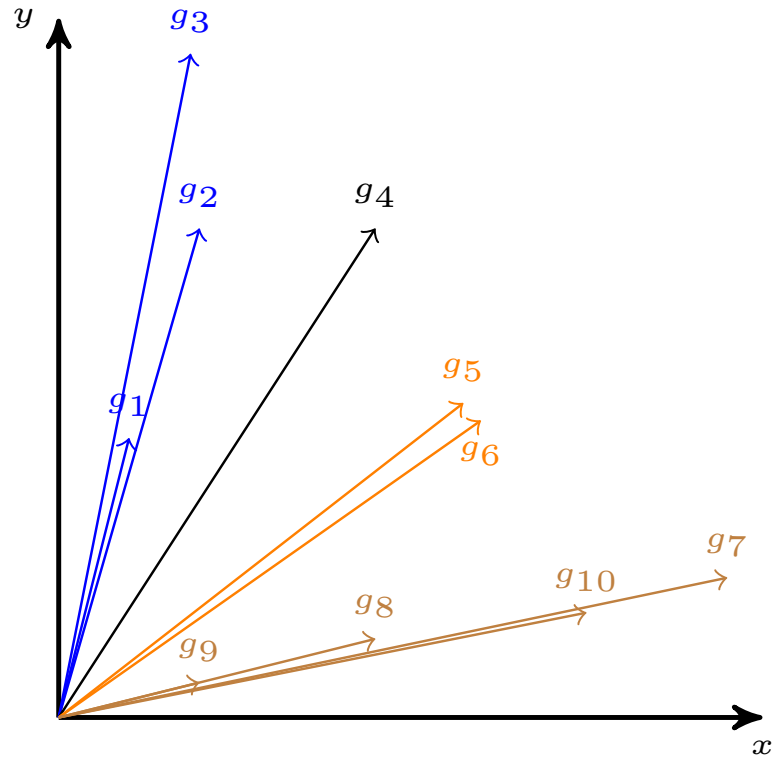
$Tl(X)$	$S(X)$	g
$false$	$false$	8
$false$	$true$	6
$true$	$false$	4
$true$	$true$	2

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

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

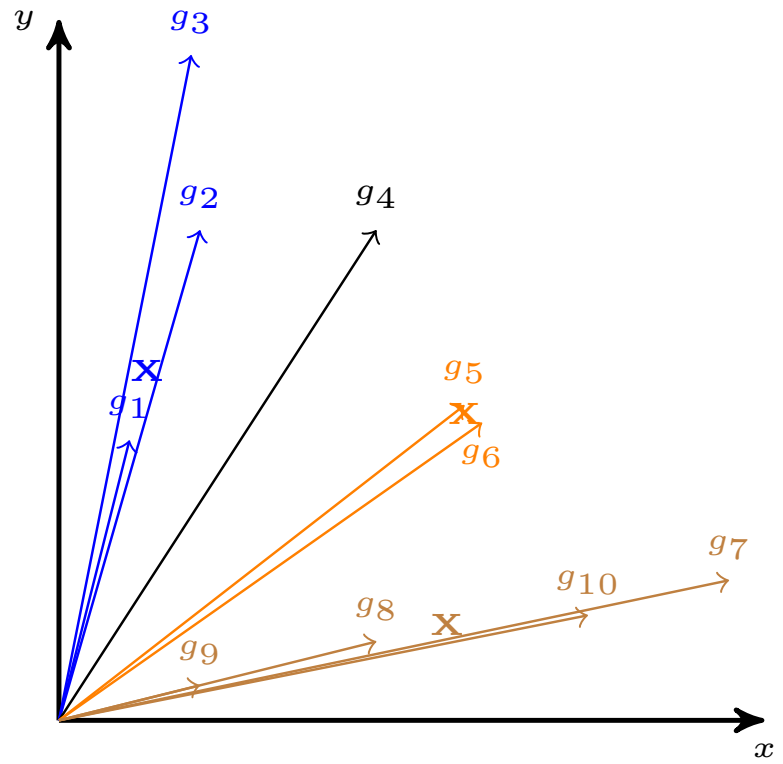
Cluster Groups

- 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?



Merge Groups

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



Merging Parfactors

- Merge similar parfactors, *accounting for groundings*

$|\mathrm{dom}(X)| = 4$

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

$|\mathrm{dom}(X')| = 4$

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

$|\mathrm{dom}(X'')| = 2$

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

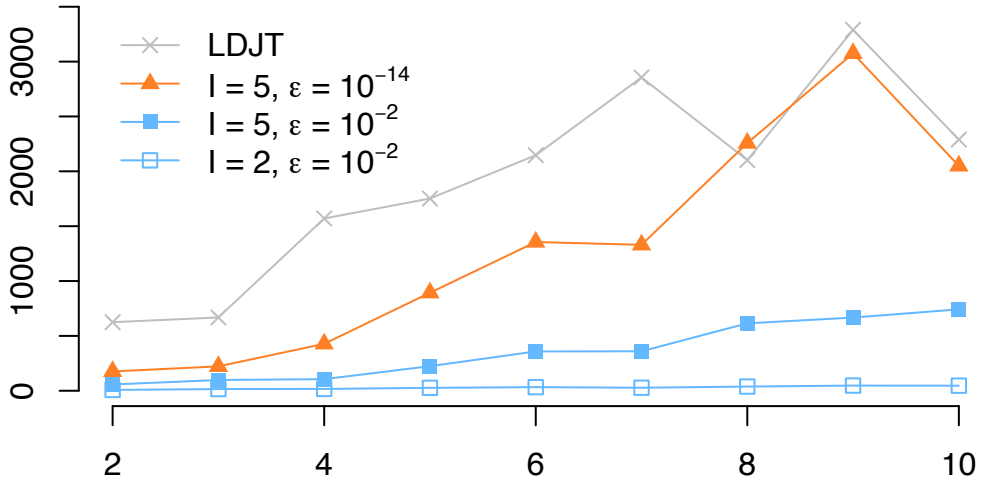
$\rightarrow$

$|\mathrm{dom}(X)| = 10$

$Tl(X)$	$S(X)$	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 (neighbourhood: ε, $n = 2$)
- ANOVA for checking fitness of clusters ($\alpha = 0.005$)
- Right: Runtimes in seconds
  - I denotes how often TAMe is run (every I'th run)
  - ε parameter for DBSCAN
- Below: prediction error of LDJT with TAMe compared to LDJT without TAMe

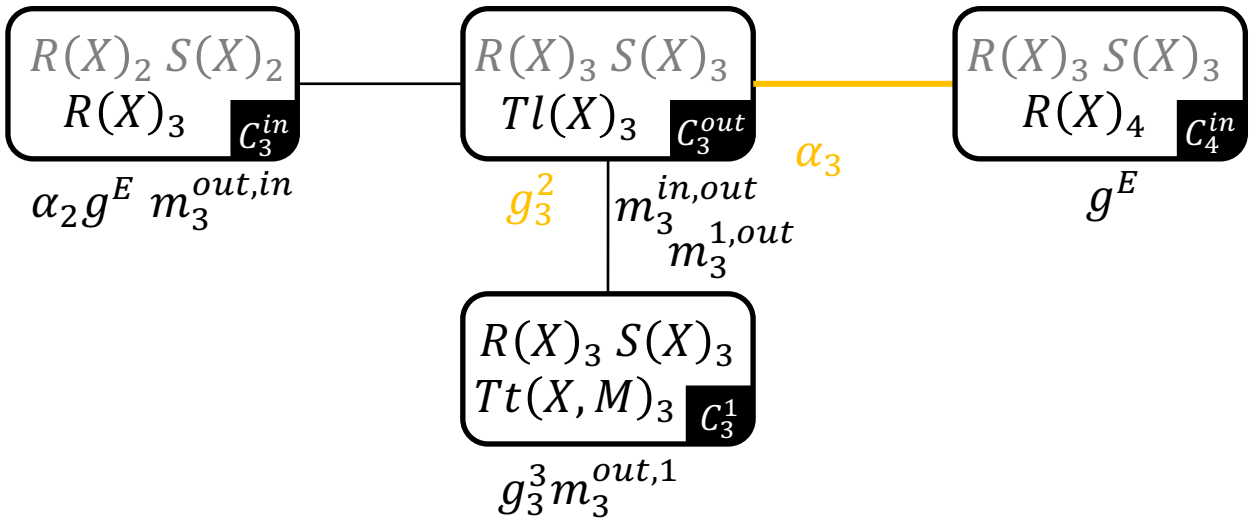


π	Max	Min	Average
0	0.0001537746121	0.0000000001720	0.0000191206488
2	0.0000000851654	0.0000000000001	0.0000000111949
4	0.0000000000478	0	0.0000000000068

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!



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

Outlook

- Changing domains over time / sequences
  - Towards open-world assumption
- Non-stationarity
  - Outside forces change the distributions
  - Maybe use ideas from 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

Outline: 6. Lifted Sequential Models & Inference

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

B. *Lifted dynamic inference*
   - Interfaces, lifted dynamic junction tree algorithm (LDJT)
   - Theoretical analysis: complexity
   - Keeping inference polynomial: Temporal approximate merging (TAMe)

⇒ Next: Decision Making