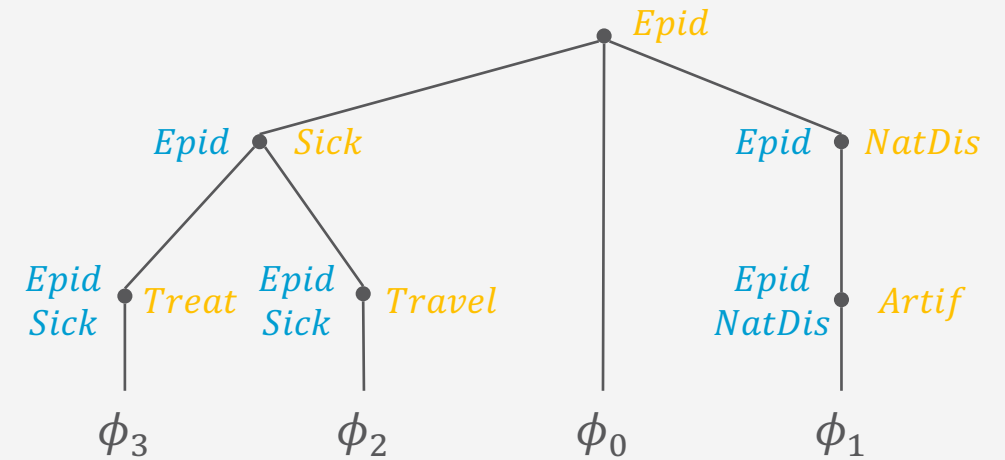


Exact Inference: Variable Elimination and the Like

Statistical Relational Artificial Intelligence
(StaRAI)



Contents

1. Introduction

- Notions of AI, agents, StaRAI

2. Modelling Formalisms

- a. Logic*
- b. Probabilistic Graphical Models (PGMs)*
- c. Probabilistic Relational Models (PRMs)*
- Syntax, semantics, inference tasks

3. Inference

- a. Exact Inference*
 - i. (Weighted) Model Counting*
 - ii. Variable Elimination*
 - iii. Lifted Inference*

b. Approximate Inference

- Inference algorithms

4. Learning

- Relation learning in logic
- Parameter / structure learning for PGMs
- From PGMs to PRMs, PRM learning

5. Sequential Models and Inference

- Syntax, semantics, inference tasks, algorithms

6. Decision-theoretic Models and Inference

- Preferences, utility
- Syntax, semantics, inference task, algorithm

Outline: 3. Inference

a. Exact Inference

i. (Weighted) Model Counting

- Model counting
- Weighted model counting, knowledge compilation (KC)

ii. Variable Elimination and the Like

- Variable elimination (VE)
- Junction tree algorithm (JT)

iii. Lifted Inference

- First-order KC, Lifted VE, lifted JT

b. Approximate Inference

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling

Inference Tasks

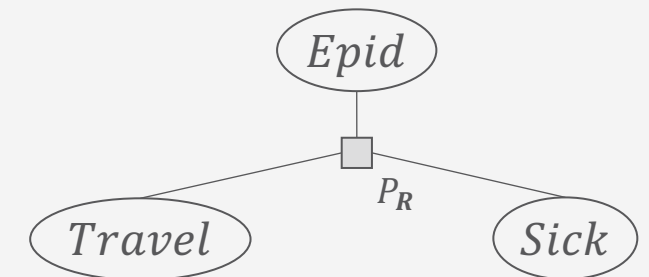
• Query Answering Problem

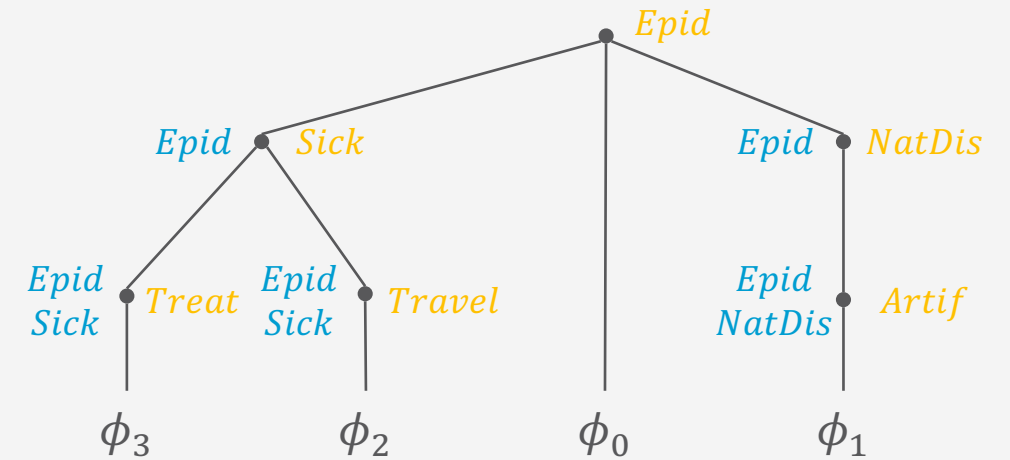
- Compute an answer to a query given full joint probability distribution P_R
 - Query for a marginal (conditional) probability (distribution)
 - Marginal probability of events: $P(epid)$
 - Marginal probability distribution of random variables: $P(Epid, Travel)$
 - Marginal **conditional** probability of events **given** random variables or events: $P(epid|sick)$
 - Marginal **conditional** probability distribution of random variables **given** random variables or events: $P(Sick|Epid)$

• Focus here: $P(S|t)$

- $rv(S, T) \subseteq R, S \cap T = \emptyset$
 - S : random variables
 - T : events t (considered observations, called **evidence**)

<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
<i>false</i>	<i>false</i>	<i>false</i>	0.20
<i>false</i>	<i>false</i>	<i>true</i>	0.24
<i>false</i>	<i>true</i>	<i>false</i>	0.28
<i>false</i>	<i>true</i>	<i>true</i>	0.08
<i>true</i>	<i>false</i>	<i>false</i>	0.05
<i>true</i>	<i>false</i>	<i>true</i>	0.06
<i>true</i>	<i>true</i>	<i>false</i>	0.07
<i>true</i>	<i>true</i>	<i>true</i>	0.02





Variable Elimination

Inference algorithm

Complexity analysis

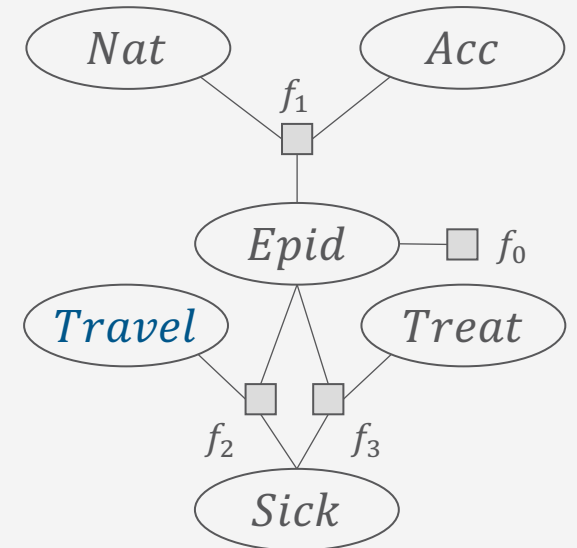
Variable Elimination (VE)

- Exact algorithm for solving query answering problems
- Exploit factorisation for efficiency
 - Push sums into product as far as possible using
 - Commutativity: Given a factor $\phi(\mathbf{R}) = \phi(\mathbf{R}_1, \mathbf{R}_2)$ with $\mathbf{R}_1 \cap \mathbf{R}_2 = \emptyset$

$$\sum_{r \in \text{ran}(\mathbf{R})} \phi(r) = \sum_{r_1 \in \text{ran}(\mathbf{R}_1)} \sum_{r_2 \in \text{ran}(\mathbf{R}_2)} \phi(r_1, r_2) = \sum_{r_2 \in \text{ran}(\mathbf{R}_2)} \sum_{r_1 \in \text{ran}(\mathbf{R}_1)} \phi(r_1, r_2)$$

- Distributivity: Given two factors $\phi(\mathbf{R}), \phi(\mathbf{S}, \mathbf{T})$ with $\mathbf{R} \cap \mathbf{S} = \emptyset$

$$\phi(\mathbf{R}) \sum_{s \in \text{ran}(\mathbf{S})} \phi(s, \mathbf{T}) = \sum_{s \in \text{ran}(\mathbf{S})} \phi(\mathbf{R}) \phi(s, \mathbf{T})$$



Variable Elimination (VE)

- Outline:

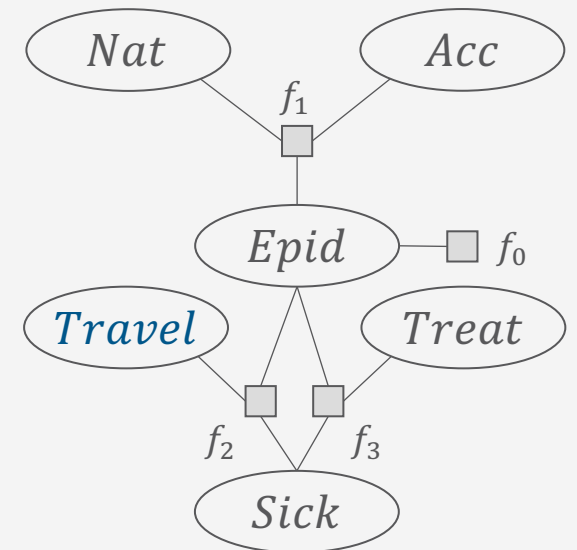
1. Absorb evidence $\mathbf{t}$ in each factor covered by $\mathbf{t}$, i.e., $\text{rv}(f) \cap \mathbf{t} \neq \emptyset$,
2. Sum out non-query variables $\mathbf{U} = \mathbf{R} \setminus \text{rv}(\mathbf{S}, \mathbf{t})$ using factorisation in model F

$$\begin{aligned}
 P(\mathbf{S} \mid \mathbf{t}) &= \frac{1}{P(\mathbf{t})} \sum_{\mathbf{u} \in \text{ran}(\mathbf{U})} P_F(\mathbf{S}, \mathbf{t}, \mathbf{U} = \mathbf{u}) \\
 &= \frac{1}{P(\mathbf{t})} \sum_{\mathbf{u} \in \text{ran}(\mathbf{U})} \prod_{f \in F} \underbrace{\phi_f(R_1, \dots, R_k)}_{\pi_{\text{rv}(f)}(\mathbf{S}, \mathbf{t}, \mathbf{U} = \mathbf{u})}
 \end{aligned}$$

- Factor out factors from sums if arguments not covered by sum

3. Divide by $P(\mathbf{t}) = \text{Normalise } P(\mathbf{S}, \mathbf{t})$

- Example: $P(\textit{Travel})$ in $F = \{f_i\}_{i=0}^3$

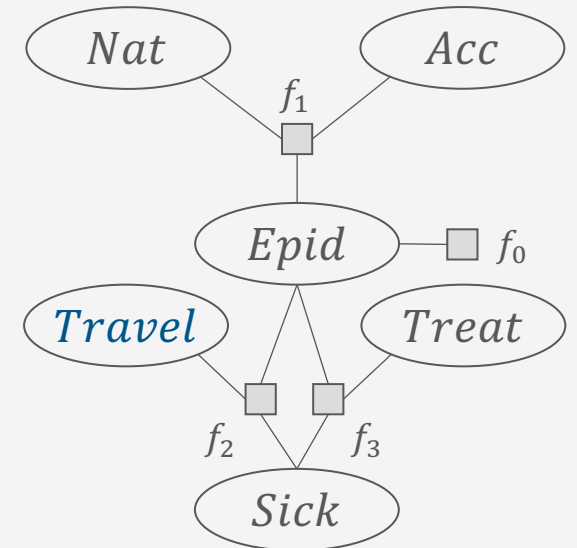


Variable Elimination (VE): Example

$P(Travel)$

$$\begin{aligned}
 &\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} P_R(E = e, N = n, A = a, S = s, Travel, T = t) \\
 &\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} \prod_{i=0}^3 \phi_i(R_i = r_i) \\
 &\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{s \in \text{Val}(S)} \sum_{t \in \text{Val}(T)} \phi_0(e) \phi_1(e, n, a) \phi_2(Travel, e, s) \phi_3(e, s, t) \\
 &\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_2(Travel, e, s) \sum_{t \in \text{Val}(T)} \phi_3(e, s, t)
 \end{aligned}$$

Sums can be computed independently → could be done in parallel

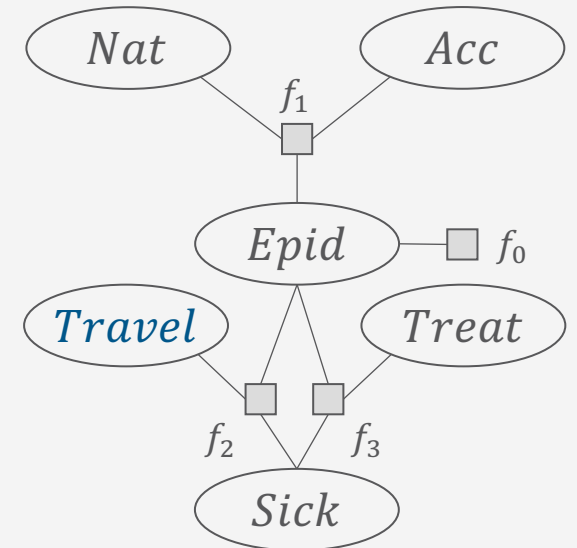
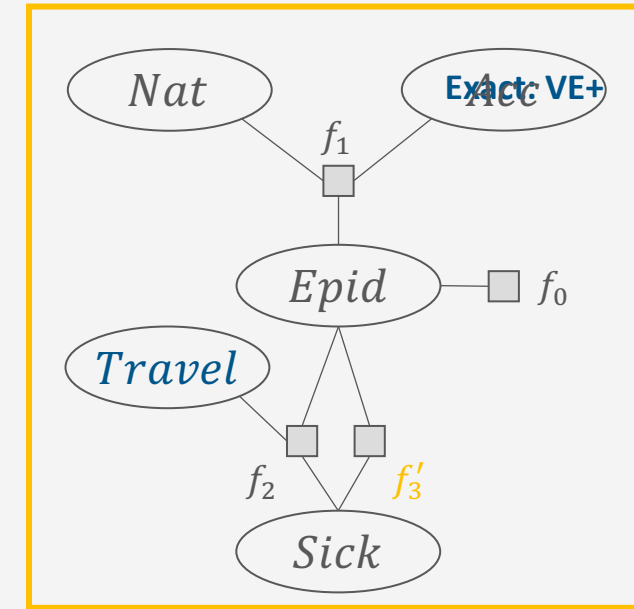


$P(\text{Travel})$

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

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_2(\text{Travel}, e, s) \phi'_3(e, s)$$

<i>Epid</i>	<i>Sick</i>	<i>Treat</i>	ϕ_3		<i>Epid</i>	<i>Sick</i>	ϕ'_3
false	false	false	5	+	false	false	6
false	false	true	1				
false	true	false	3	+	false	true	5
false	true	true	2				
true	false	false	5	+	true	false	9
true	false	true	4				
true	true	false	1	+	true	true	8
true	true	true	7				



$P(Travel)$

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

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_2(Travel, e, s) \phi'_3(e, s)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_{23}(Travel, e, s)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(Travel, e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a)$$

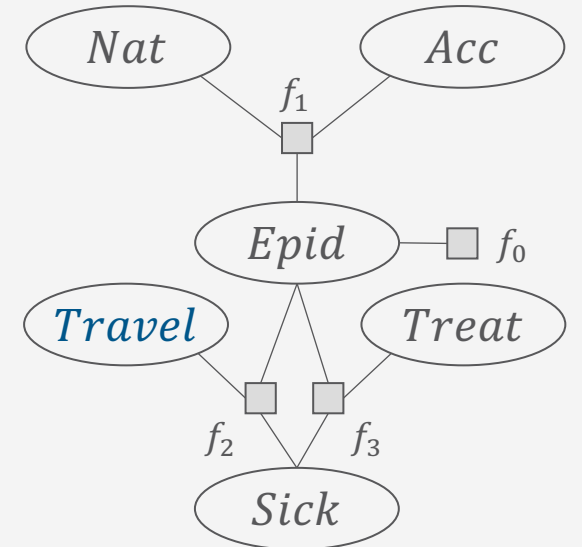
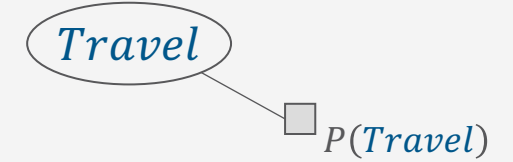
$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(Travel, e) \phi''_1(e) = \sum_{e \in \text{Val}(E)} \phi(Travel, e)$$

$$= \phi'(Travel)$$

$$= \phi^n(Travel) = P(Travel)$$

Intermediate results never larger than $2^3 < 2^6$

Exact: VE+



VE with Evidence

- Absorb each observation $t \in \mathbf{t}$ in each factor f , $t \cap rv(f) \neq \emptyset$
 - Drop rows with $T \neq t \rightarrow$ select all row with $T = t : f \leftarrow \sigma_{T=t}(f)$
 - Drop $T \rightarrow$ project result onto $rv(f) \setminus \{T\} : f \leftarrow \pi_{rv(f) \setminus \{T\}}(f)$
- Example: $P(\text{Travel}|\text{sick}) \rightarrow f_2, f_3$ have to absorb *sick*

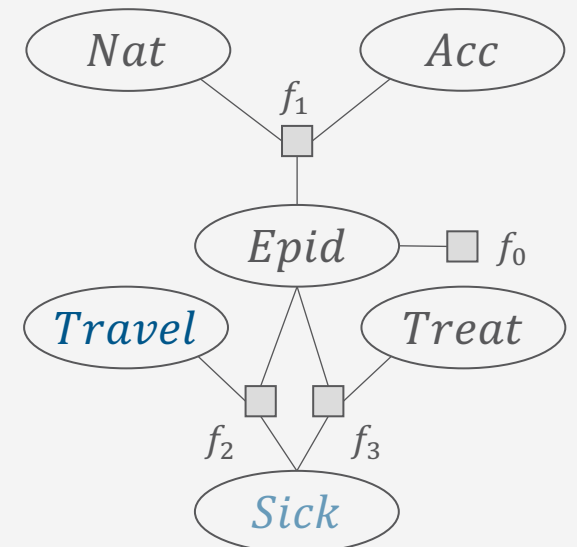
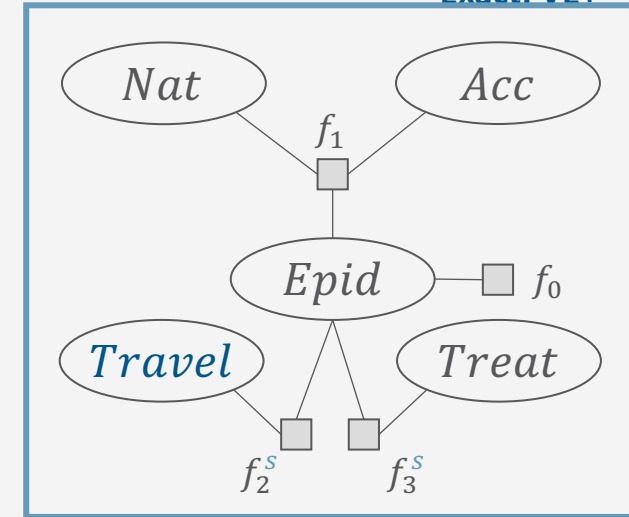
Epid	Sick	Treat	ϕ_3
false	false	false	5
false	false	true	1
false	true	false	3
false	true	true	2
true	false	false	5
true	false	true	4
true	true	false	1
true	true	true	7

Epid	Treat	ϕ_3^S
false	false	3
false	true	2
true	false	1
true	true	7

Travel	Epid	Sick	ϕ_2
false	false	false	20
false	false	true	24
false	true	false	5
false	true	true	6
true	false	false	28
true	false	true	8
true	true	false	7
true	true	true	2

Travel	Epid	ϕ_2^S
false	false	24
false	true	6
true	false	8
true	true	2

Exact: VE+



VE with Evidence: Example

$P(\textit{Travel}|\textit{sick})$

$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{t \in \text{Val}(T)} P_R(E = e, N = n, A = a, \textit{sick}, \textit{Travel}, T = t)$$

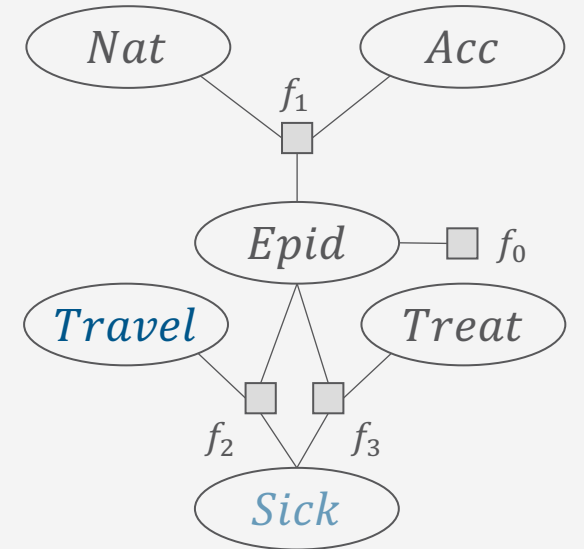
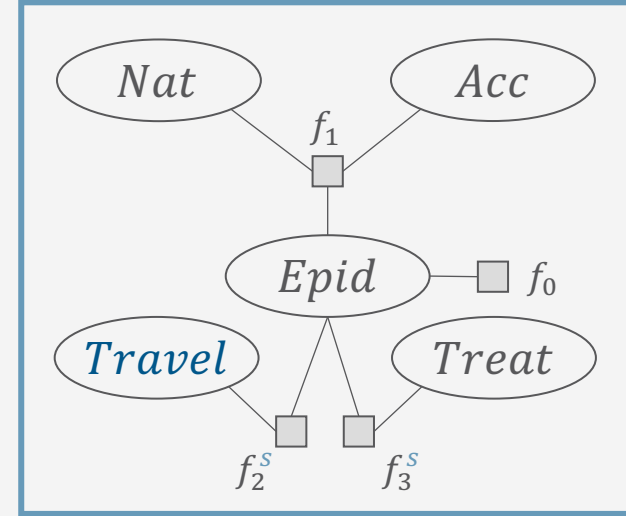
$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{t \in \text{Val}(T)} \prod_{i=0}^3 \phi_i(R_i = r_i)$$

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

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

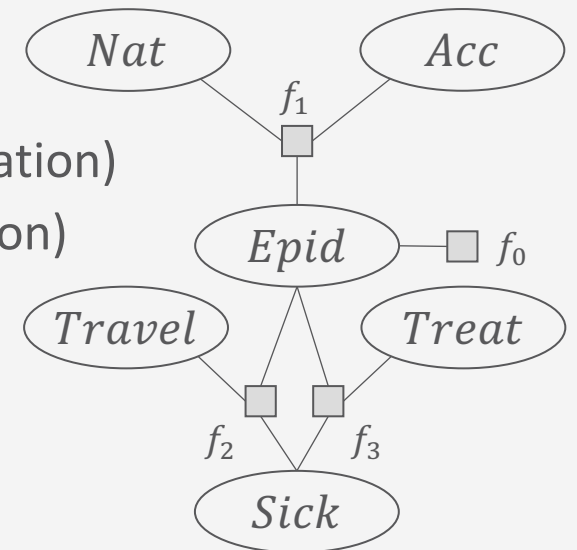
$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi_2^S(\textit{Travel}, e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{t \in \text{Val}(T)} \phi_3^S(e, t)$$

Exact: VE+



Elimination Order & Complexity

- Elimination order important
 - Wrong order $\rightarrow$ large intermediate result: consider eliminating *Epid* first
 - Finding the best order not easier than inference using full joint
 - A lot of research has gone into finding a good order
- *Online greedy heuristic*: Variable R with smallest intermediate result
 - For each possible R to sum out
 - Collect all factors F_R containing R (would need to be multiplied before elimination)
 - Take number of arguments $|rv(F_R)|$ (Intermediate result size before elimination)
 - Decision criterion: $\arg \min_R |rv(F_R)| \rightarrow$ one-step VE simulation

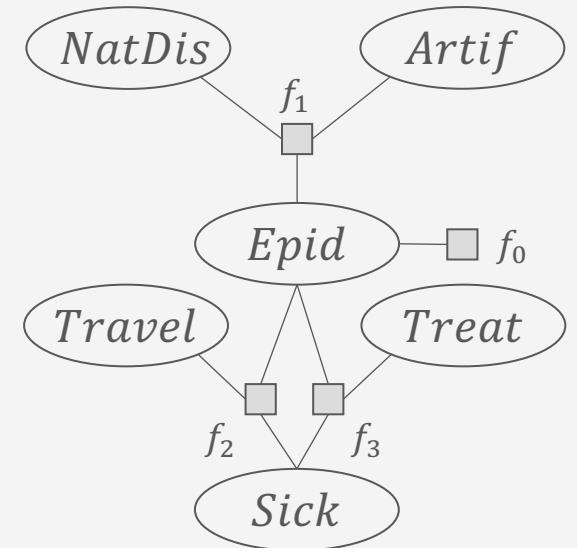
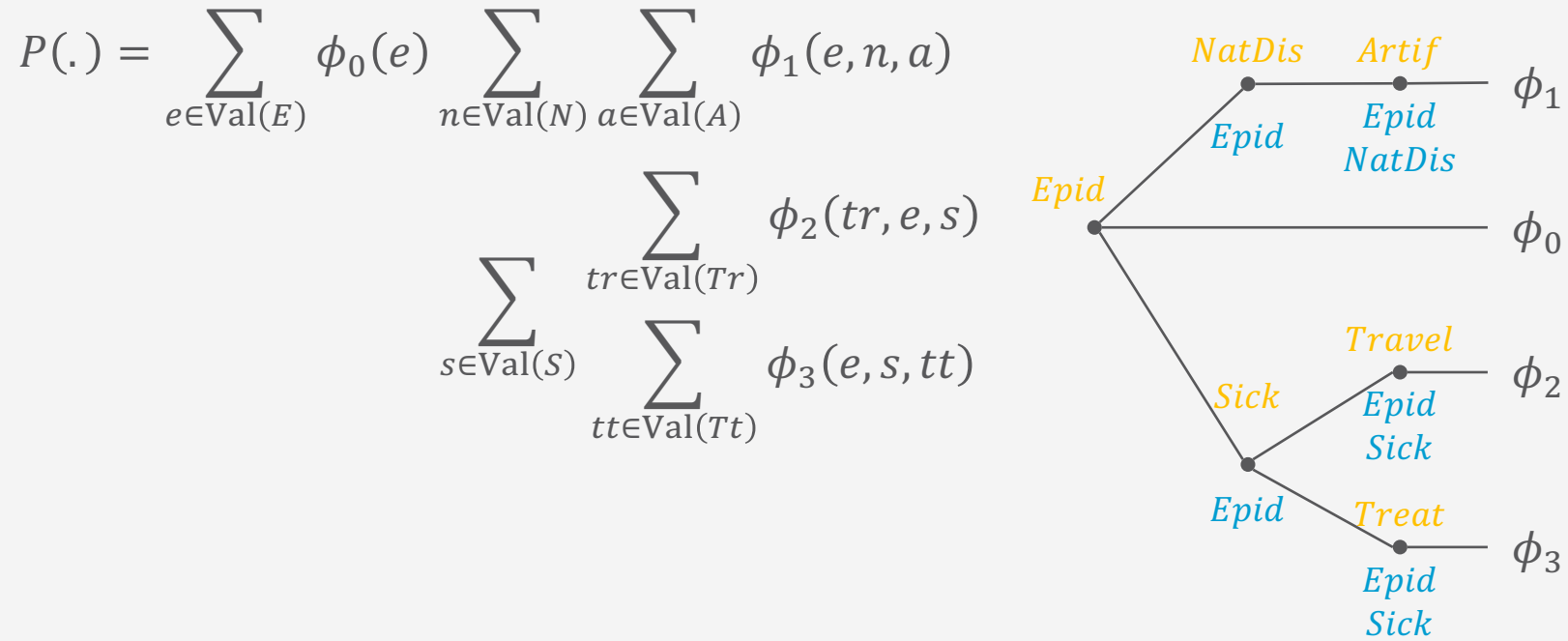


Runtime Complexity of Probabilistic Inference in PGMs

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

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

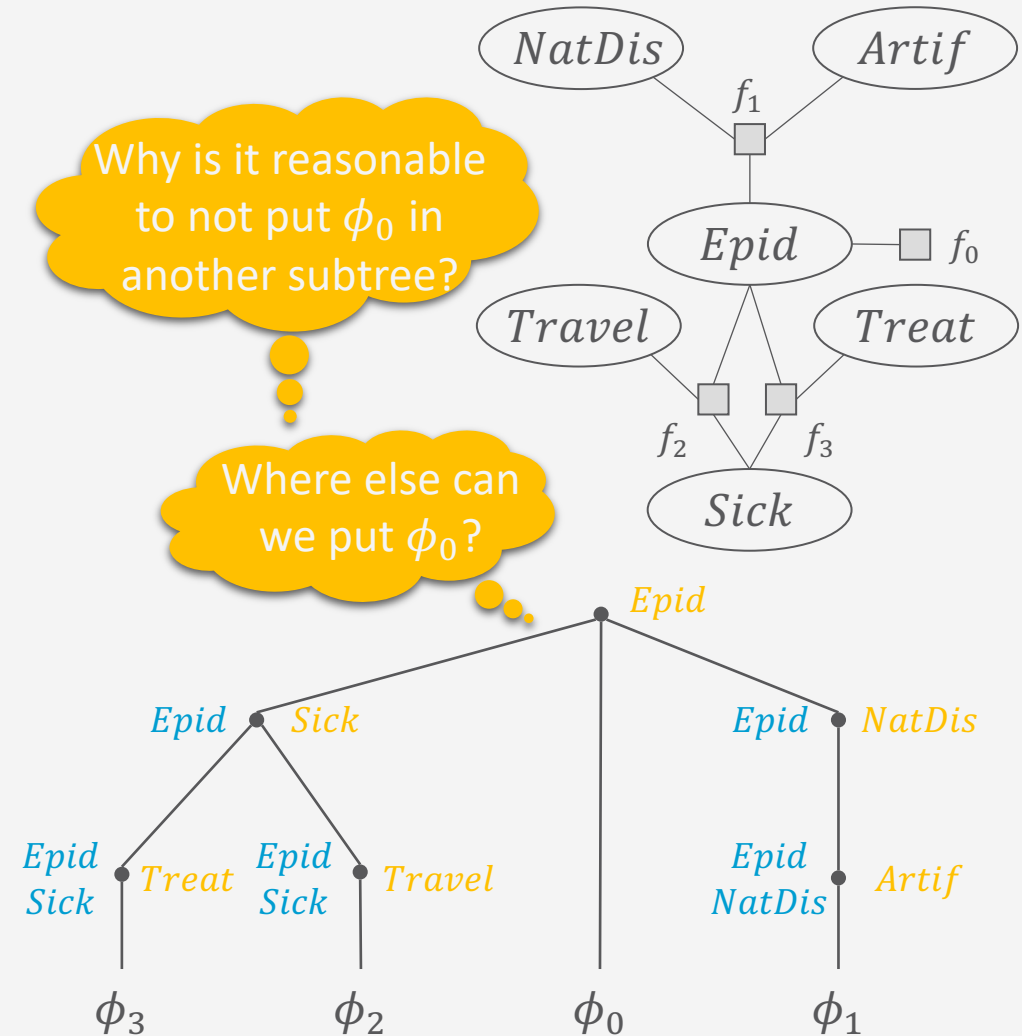
Example: Dtree – Bottom-up Construction as VE Representation



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

Example: Dtree – Top-down Interpretation

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



Cutset, Context, Cluster

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

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

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

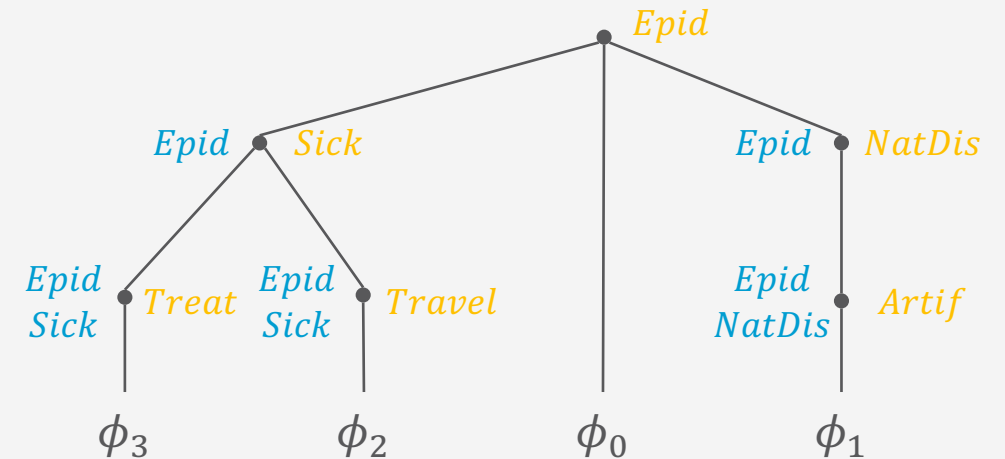
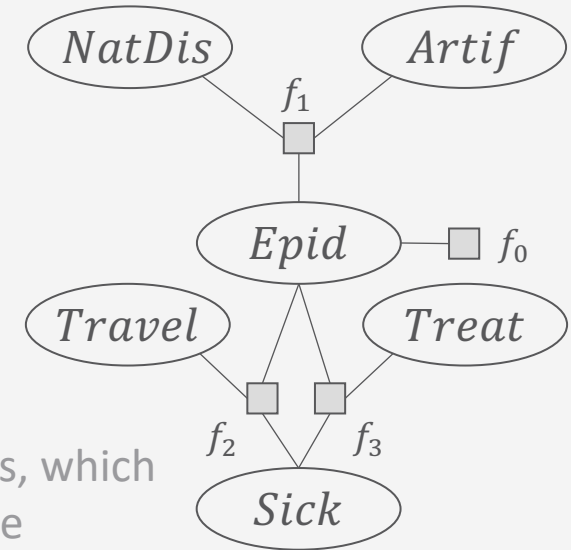
$\text{rv}(T)$ returns the random variables, which appear in the factors of this subtree

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

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

- Cluster
 - Cutset and context together

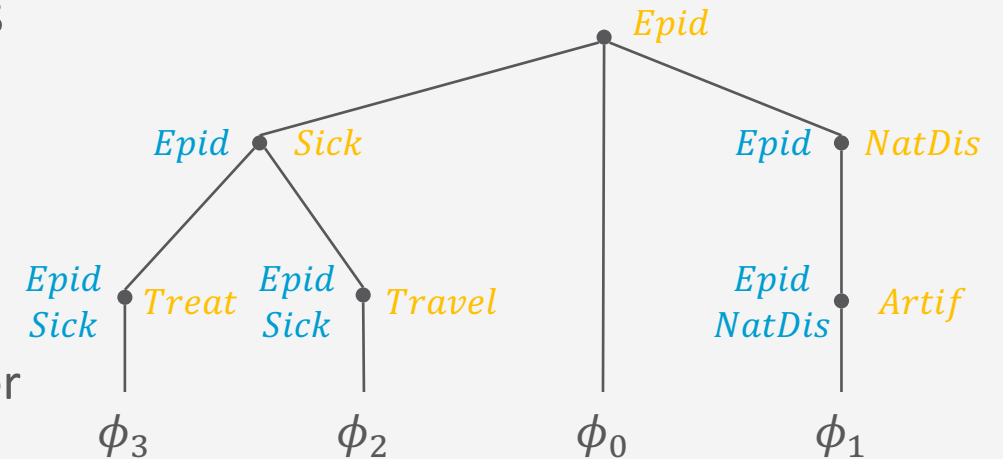
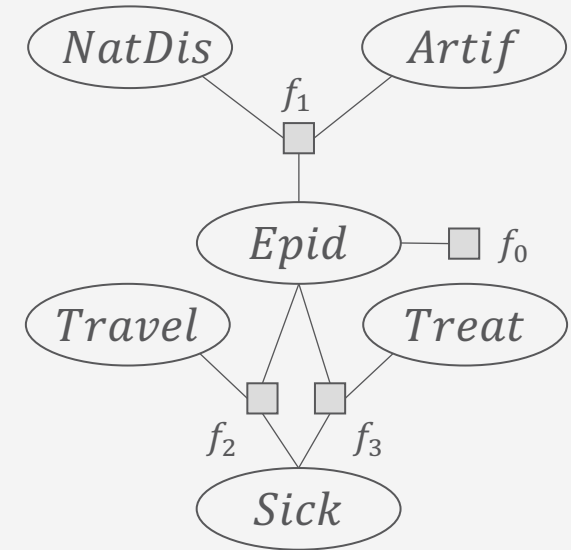
$$\text{cluster}(T) = \text{cutset}(T) \cup \text{context}(T)$$
 - If T is a leaf, then $\text{cluster}(T) = \text{context}(T) = \text{rv}(f)$



Cutset, Context, Cluster

- Largest cluster in tree $T = \text{width}$

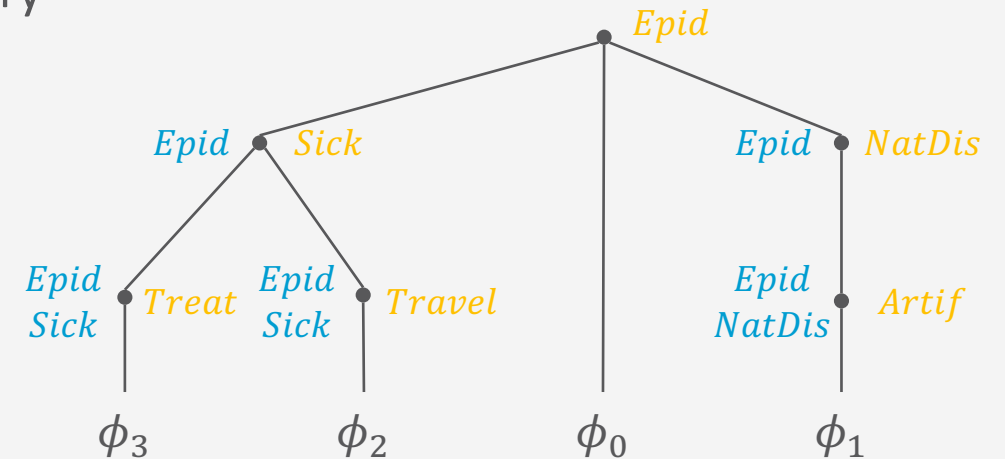
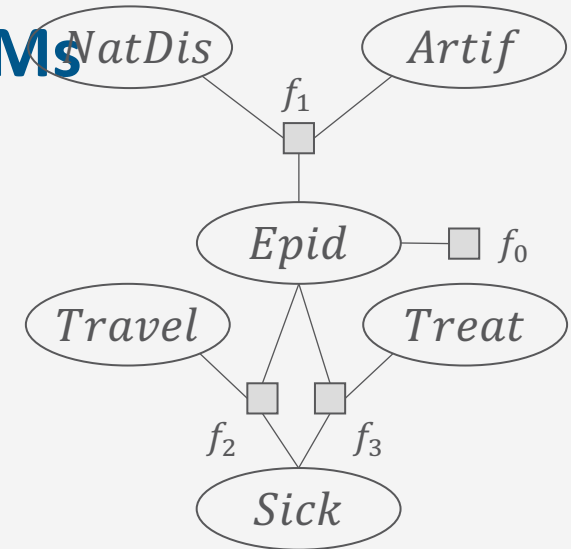
$$\text{width}(T) = \max_{T_i \in \text{Desc}(T)} |\text{cluster}(T_i)|$$
 - Bounded from below by largest input factor size: $w \geq m = \max_{f \in F} |\text{rv}(f)|$
 - When learning a model, avoid factors with many arguments (e.g., bound degree in FG / MN)
- **Tree width** w = minimal width over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



Back to Runtime Complexity of Probabilistic Inference in PGMs

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

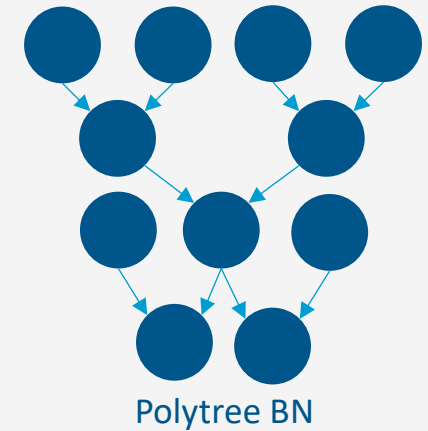
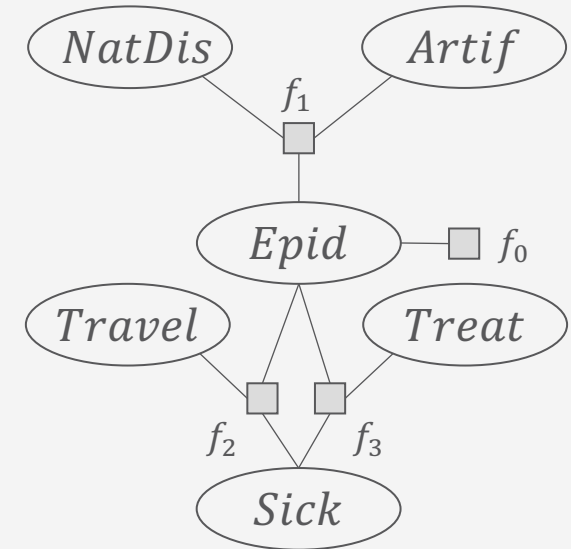
$$O(n_T \cdot r^w)$$
 - $r = \max_{R \in rv(F)} |ran(R)|$
 - Compare with inference using full joint P_F : $O(r^{n_T})$



Complexity and Tractability

- Query answering problem is **tractable IF**
 - it is solved by an efficient algorithm in **time polynomial w.r.t. the number of random variables**
- Query answering problem in general is **intractable**
 - No guarantees that $w \ll n_T$
- Exceptions make assumptions about model structure
 - E.g., *polytree* BNs B
 - Polytree: no cycles in underlying undirected graph
 - Ensures that $w = \max_{R \in \text{rv}(B)} |pa(R)| + 1$
 - Also holds for tree-shaped FGs and their MN representation
 - Assumes that degree is not in order of n_T

Exact: VE+

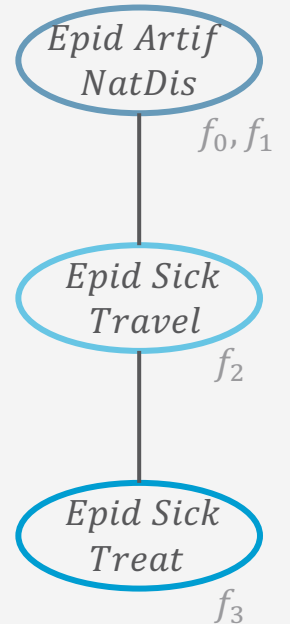


Interim Summary

- Variable elimination (VE): inference algorithm to solve query answering problems
 - Absorb evidence, multiply factors, sum-out variables
 - Good elimination order required (heuristics)
- Complexity possibly reduced to $O(N \cdot r^w)$
 - Decomposition tree allows for analysis of runtime complexity
 - Cutset, context, cluster
 - Largest cluster size under optimal elimination order $\rightarrow$ tree width
- Problem: VE restarts from scratch with each new query

Junction Tree Algorithm

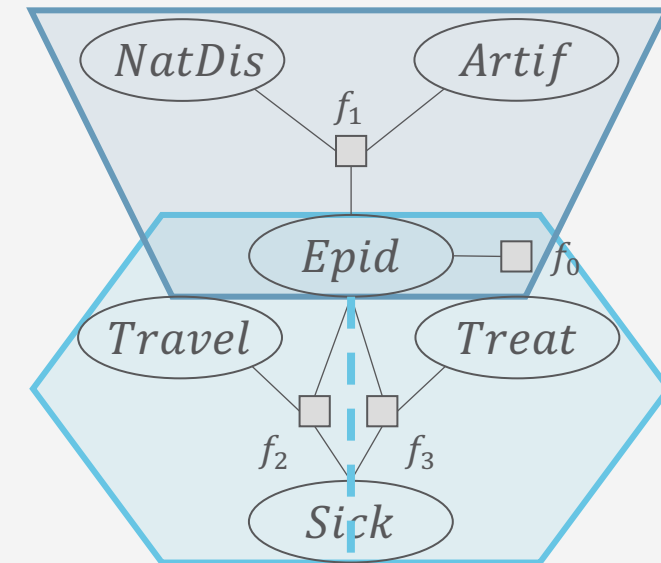
Multi-query inference algorithm



Example

- Set of queries
 - $P(Epid), P(NatDis), P(Artif), P(Travel), P(Treat), P(Sick)$
 - For $P(NatDis), P(Artif)$, the *lower* part of the model ...
 - For $P(Travel), P(Treat), P(Sick)$, the *upper* part of the model ...
- ... would be summed out in the same way
- Would it not be nice if for the queries
 - $P(NatDis), P(Artif)$ only the *upper* part of the model ...
 - $P(Travel), P(Treat), P(Sick)$ only the *lower* (left / right) part of the model ...
- ... would be relevant and the rest is available beforehand?
- Global independences to the rescue!

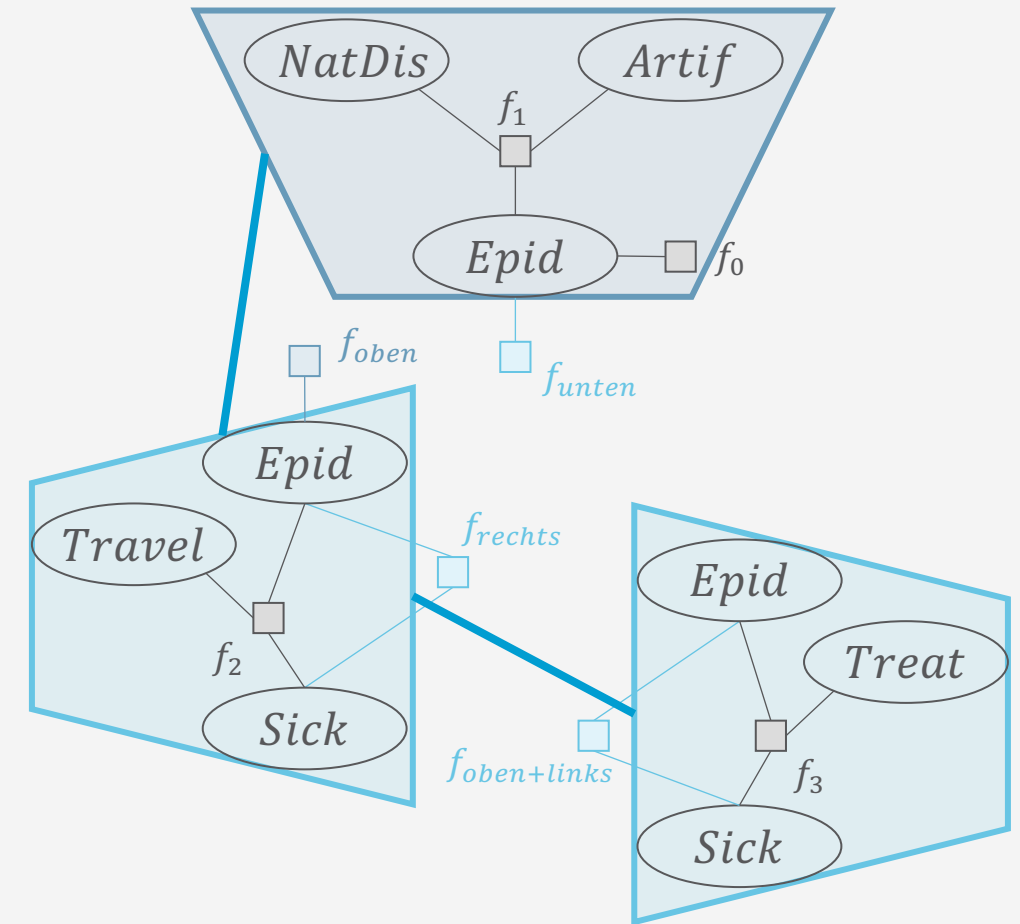
$P(Epid)$ could be considered as part of the upper or lower part



Network of Clusters

- Build a network of cluster with global independences on the shared variables between them → form separators
- Exchange information behind neighbouring clusters over those separators
- Send messages in two phases:
 - From one end to the other and back, i.e., two messages for each edge in the network
 - For maximum parallelisability: Send messages from periphery to centre and back

Idea of the Junction Tree Algorithmus



Junction Trees

- Network of clusters called **junction tree** (jtree)
 - Sometimes also referred to as *clique tree* (PGM book), *jointree* (Darwiche bool), Markov trees
- Formal definition: **jtree** $J = (V, E)$
 - Acyclic undirected graph
 - Nodes are sets of random variables, called **Clusters**
 - Has to fulfil three properties for a model $F = \{f_i\}_{i=1}^n$ (otherwise not a jtree):
 1. Each cluster C contains random variables of F : $\forall C \in V : C \subseteq \text{rv}(F)$
 2. For each factor, its inputs appear in the same cluster: $\forall f \in F : \text{rv}(f) \subseteq C$
 3. [**Running intersection property**] If a random variable appears in two clusters, then it appears in each cluster on the path between the two clusters:
$$(\exists R \in \text{rv}(F) : R \in C_i \wedge R \in C_j \wedge C_i, C_j \in V) \Rightarrow (\forall C_k \in \text{path}(C_i, C_j) : R \in C_k)$$
 - **Minimal**: If no random variable can be removed from any cluster without J ceasing to be a jtree for F , i.e., at least one of the three jtree properties no longer holds, then J is minimal

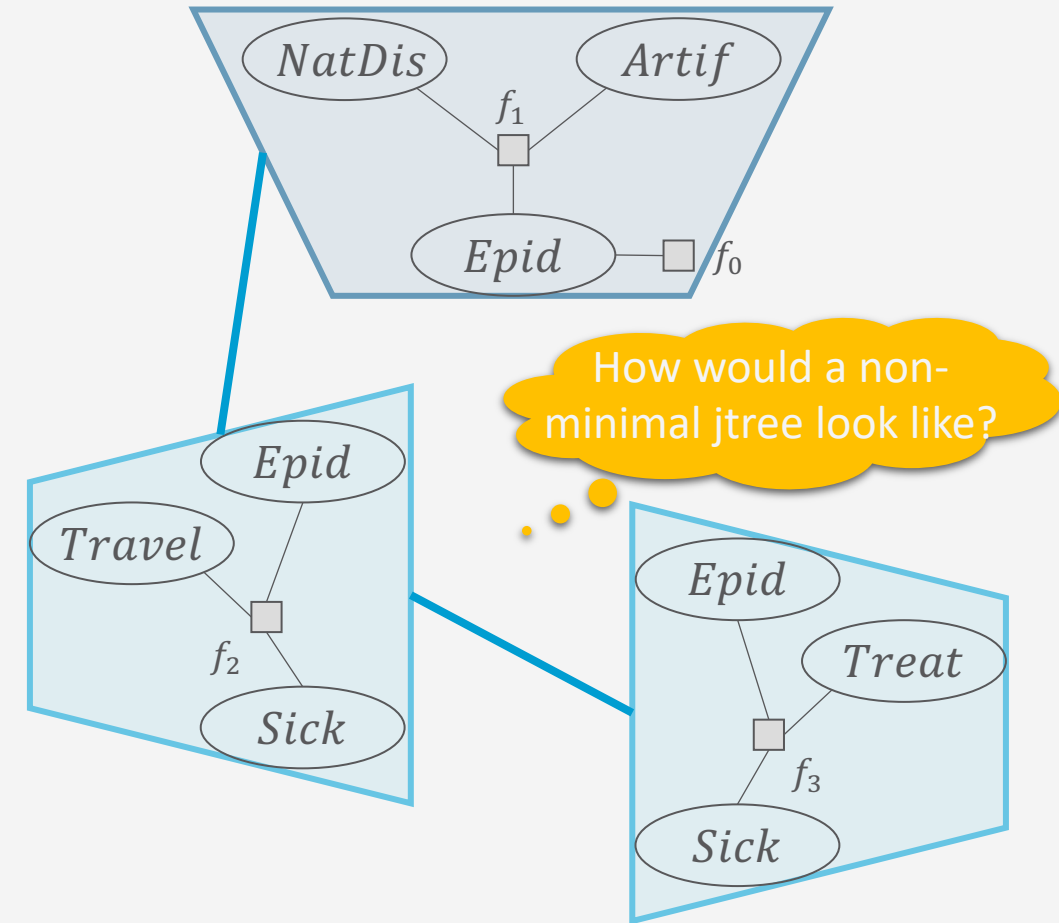
Jtree Properties in the Example

1. Each cluster $\mathcal{C}$ contains random variables of F :

$$\forall \mathcal{C} \in V : \mathcal{C} \subseteq \text{rv}(F)$$
2. For each factor, its inputs appear in the same cluster:

$$\forall f \in F : \text{rv}(f) \subseteq \mathcal{C}$$
3. [*Running intersection property*] If a random variable appears in two clusters, then it appears in each cluster on the path between the two clusters:

$$(\exists R \in \text{rv}(F) : R \in \mathcal{C}_i \wedge R \in \mathcal{C}_j \wedge \mathcal{C}_i, \mathcal{C}_j \in V) \\ \Rightarrow (\forall \mathcal{C}_k \in \text{path}(\mathcal{C}_i, \mathcal{C}_j) : R \in \mathcal{C}_k)$$
- Example:
 1. Each cluster contains random variables from the model
 2. Each factor is covered by a cluster
 3. Variable *Epid* appears in $\mathcal{C}_{upper}$, $\mathcal{C}_{right}$ and on the path between them, namely, in $\mathcal{C}_{left}$
 - Minimal, or else 2. would no longer hold

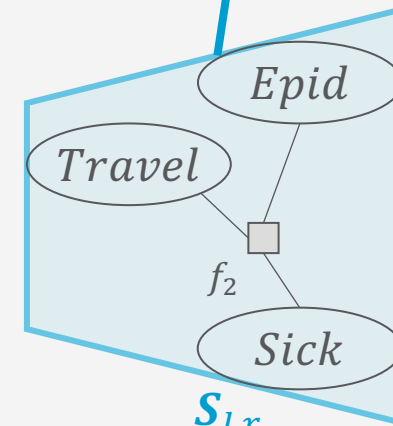
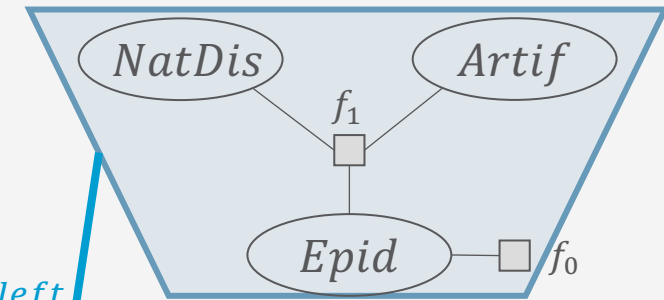


Jtrees

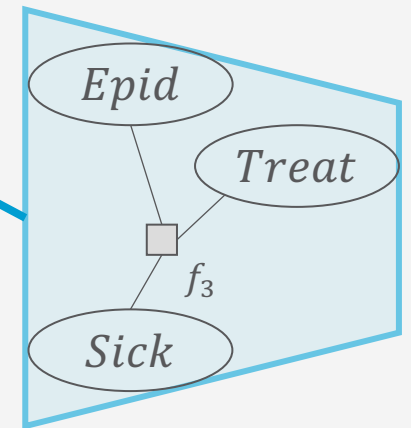
- Given a model F with a jtree $J = (V, E)$
- Separator S_{ij}
 - Set of random variables that appears in the neighbouring clusters C_i, C_j
 - Formally, $S_{ij} = C_i \cap C_j$
- Local model F_i of a cluster C_i
 - Assign each factor to a cluster
 - It holds that $\forall f \in F_i : \text{rv}(f) \subseteq C_i$
 - The F_i partition $F: F = \bigcup_{C_i \in V} F_i, F_i \cap F_j = \emptyset$
 - Example: $F_{upper} = \{f_0, f_1\}, F_{left} = \{f_2\}, F_{right} = \{f_3\}$

Unique
assignment?

$$\begin{aligned} S_{u,l} &= C_{upper} \cap C_{left} \\ &= \{Epid\} \end{aligned}$$



$$\begin{aligned} S_{l,r} &= C_{left} \cap C_{right} \\ &= \{Epid, Sick\} \end{aligned}$$

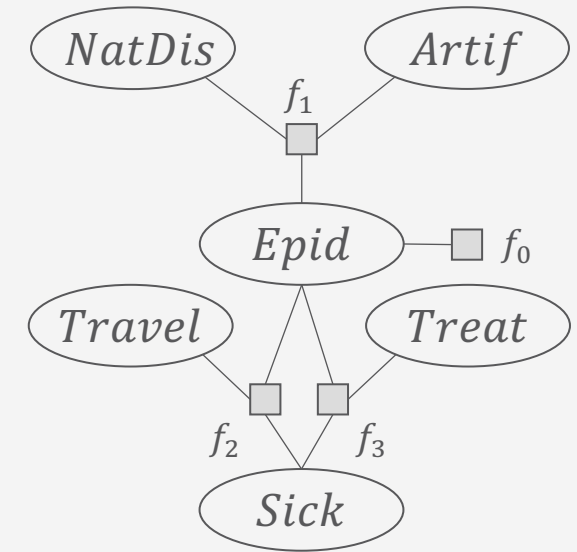


Junction Tree Algorithm (JT)

- Input
 - Model F
 - Evidence e
 - Query variables $\{Q_i\}_{i=1}^m$
 - Queries asked for $F: P(Q_i \mid e)$ for $i \in \{1, \dots, m\}$
- JT consists of four steps:
 1. Build a jtree J for model F
 2. Enter evidence e in J
 3. Pass messages in J
 4. Answer queries $\{Q_i\}_{i=1}^m$

Offline preprocessing

 - Independent of $\{Q_i\}_{i=1}^m$



Evidence:
sick

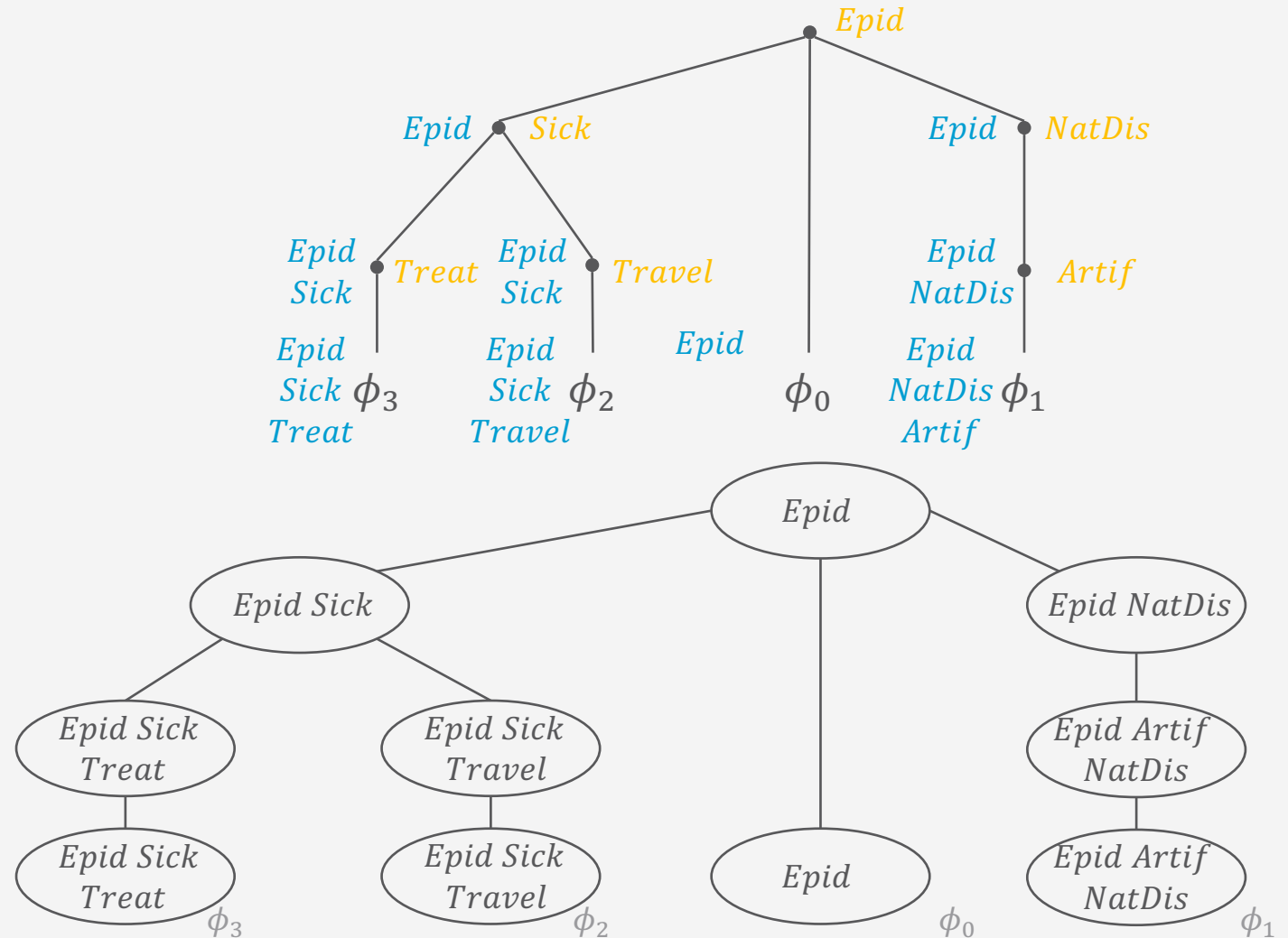
Query variables:
 $\{\{Travel\}, \{Epid\}, \{Travel, Treat\}\}$

Construct Jtrees

- With the help of dtrees:
 - Clusters of a dtree form a non-minimal jtree
 - [Without proof] Darwiche, 2001

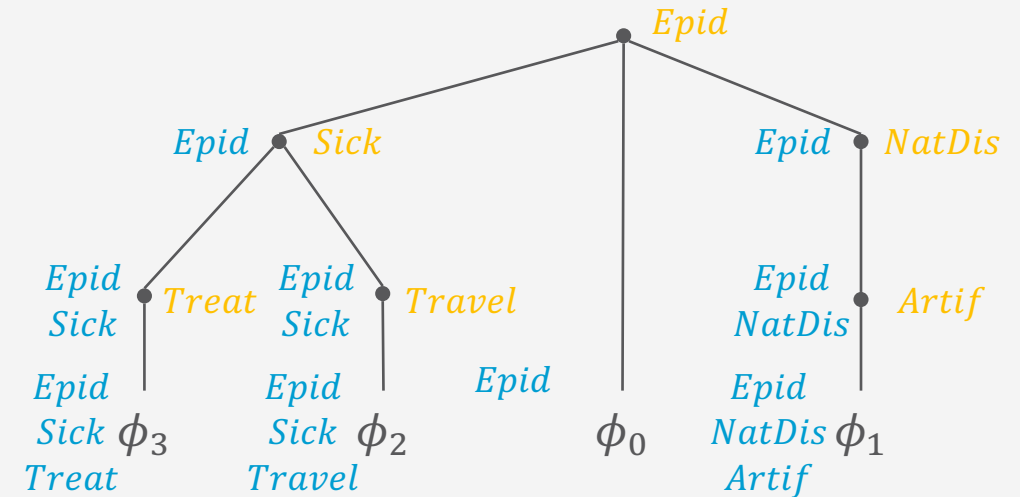
1. Each cluster $\mathcal{C}$ contains random variables of F : $\forall \mathcal{C} \in V : \mathcal{C} \subseteq \text{rv}(F)$
2. For each factor, its inputs appear in the same cluster: $\forall f \in F : \text{rv}(f) \subseteq \mathcal{C}$
3. [Running intersection property] If a random variable appears in two clusters, then it appears in each cluster on the path between the two clusters:

$$(\exists R \in \text{rv}(F) : R \in \mathcal{C}_i \wedge R \in \mathcal{C}_j \wedge \mathcal{C}_i, \mathcal{C}_j \in V) \\
 \Rightarrow (\forall \mathcal{C}_k \in \text{path}(\mathcal{C}_i, \mathcal{C}_j) : R \in \mathcal{C}_k)$$



Construct Jtrees

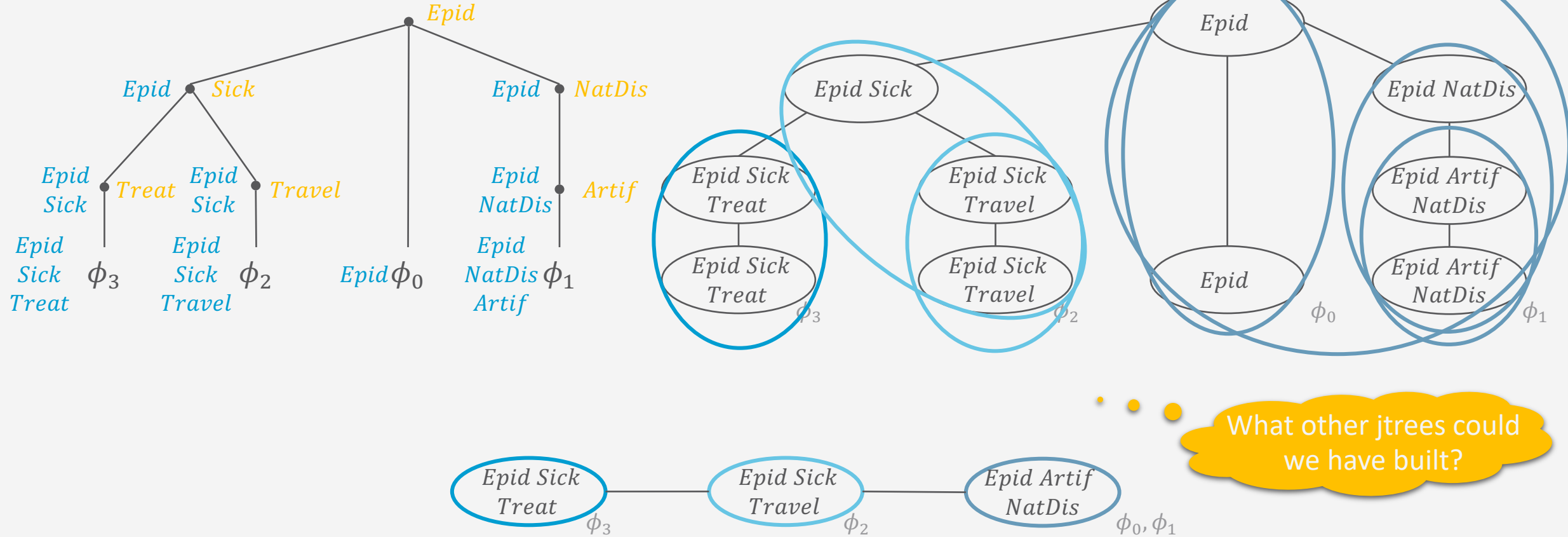
- With the help of dtrees:
 - Clusters of a dtree form a non-minimal jtree
 - [Without proof] Darwiche, 2001
 - Construct jtree



1. Construct dtree
 - Bottom up: Search for elimination order and construct dtree
 - Alternative top down approach: Partition model using a heuristic → *Hypergraph Partitioning*
2. Determine clusters in dtree using cutset and context
3. Construct a non-minimal jtree as a copy of the dtree with undirected edges and clusters as nodes, take over factors as local models in leaves
4. Minimise jtree
 - Merge clusters that are subsets of neighbours into neighbour, including local models

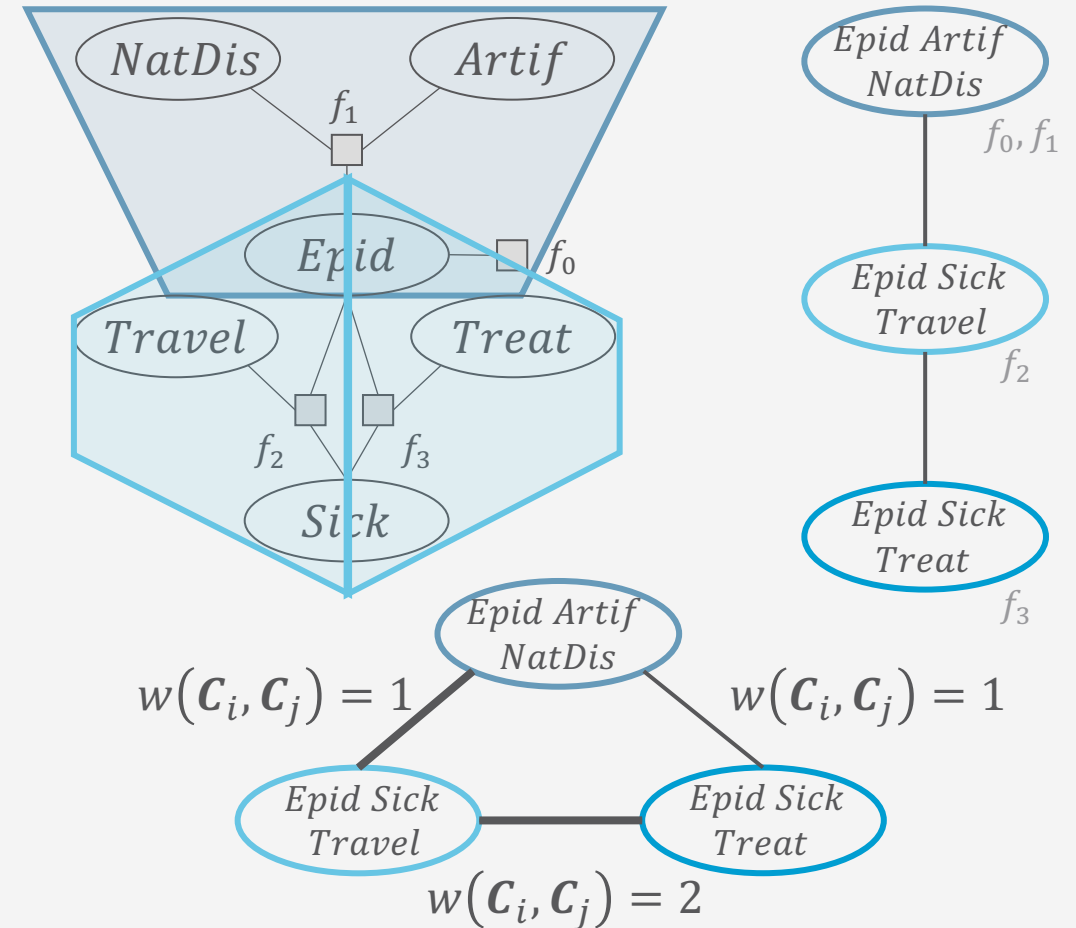
JT Step 1

Construct Jtrees: Example



Construct Jtrees: An Alternative

- Triangulation + Max-ST (*maximum spanning tree*)
 1. Triangulate graph $\rightarrow$ chordal graph
 2. Identify maximal cliques
 3. Build an undirected graph with maximal cliques as nodes and edges weighted with $w(\mathbf{C}_i, \mathbf{C}_j) = |\mathbf{C}_i \cap \mathbf{C}_j|$ whenever $w(\mathbf{C}_i, \mathbf{C}_j) > 0$
 4. Construct a maximum spanning tree based on $w(\mathbf{C}_i, \mathbf{C}_j) \rightarrow$ jtree
 5. Assign each factor to a fitting cluster
 - If BN as input: Moralise BN first
 - Does not need an elimination order but requires triangulating the graph
 - Equally hard problem



Evidence in JT

- Result after Step 1: Jtree $J = (V, E)$ with local models F_i from F
- Given evidence e encoded as factors $\{\phi_e(E)\}_{e \in e}$

Two options

1. For each $f_e = \phi_e(E) \in \{\phi_e(E)\}_{e \in e}$
 - For each $C_i \in V$
 - If $E \in C_i$, then for each $f \in F_i$ with $E \in \text{rv}(f)$
 - $F_i \leftarrow F_i \setminus \{f\} \cup \text{ABSORB}(f, E, f_e)$
2. For each $f_e = \phi_e(E) \in \{\phi_e(E)\}_{e \in e}$
 - Find a $C_i \in V$ with $E \in C_i$
 - Choose an $f \in F_i$ with $E \in \text{rv}(f)$
 - $F_i \leftarrow F_i \setminus \{f\} \cup \text{MULTIPLY}(f, f_e)$

$Sick \notin C_1$

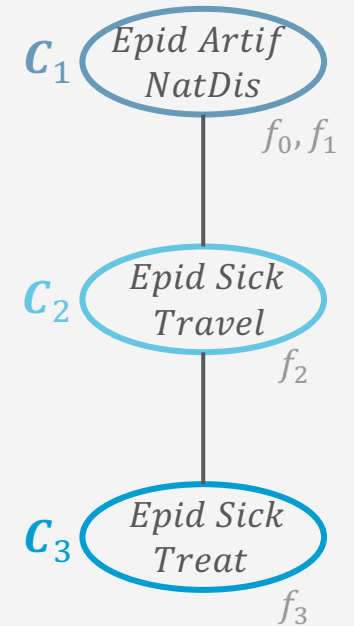
$Sick \in C_2 \rightarrow f_2^S = \text{ABSORB}(f_2, Sick, f_s)$

$Sick \in C_3 \rightarrow f_3^S = \text{ABSORB}(f_3, Sick, f_s)$

$Sick \in C_2 \rightarrow$
 $f_2^0 = \text{MULTIPLY}(f_2, f_s)$

Travel	Epid	Sick	ϕ_2
false	false	false	0
false	false	true	24
false	true	false	0
false	true	true	6
true	false	false	0
true	false	true	8
true	true	false	0
true	true	true	2

$sick \rightarrow \phi_s(Sick)$



Evidence in JT

- Result after Step 1: Jtree $J = (V, E)$ with local models F_i from F
- Given evidence e encoded as factors $\{\phi_e(E)\}_{e \in e}$

Two options

1. For each $f_e = \phi_e(E) \in \{\phi_e(E)\}_{e \in e}$ JT Step 2

- For each $C_i \in V$
 - If $E \in C_i$, then for each $f \in F_i$ with $E \in \text{rv}(f)$
 - $F_i \leftarrow F_i \setminus \{f\} \cup \text{ABSORB}(f, E, f_e)$

2. For each $f_e = \phi_e(E) \in \{\phi_e(E)\}_{e \in e}$

- Find a $C_i \in V$ with $E \in C_i$
- Choose an $f \in F_i$ with $E \in \text{rv}(f)$
- $F_i \leftarrow F_i \setminus \{f\} \cup \text{MULTIPLY}(f, f_e)$

We take Option 1

Absorb evidence in every factor with VE

- Each cluster is tested ($E \in C_i$)
- For each cluster with $E \in C_i$, each factor is tested (if $E \notin C_i$, then not in F_i)
- Dimension reduction um $|e|$, da e eliminiert wird

Multiply with $\langle 1, 0 \rangle$ -factor, 0 is later propagated

- “Find” and “Choose” some worst-case effort as above, but multiplication done only once
- But, no dimension reduction $\rightarrow E$ has to be summed out later

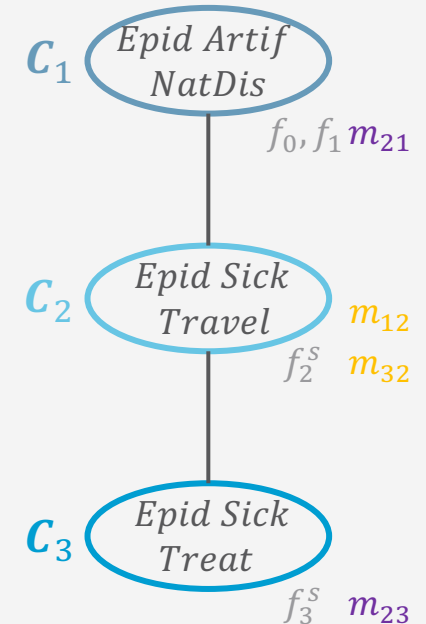
Message Passing

- Result after Step 2: Local models updated with evidence
- Two conditions to send messages
 1. If cluster C_i has received all messages except the one from neighbour C_j , C_i sends message m_{ij} to C_j .
 - Always true at leaves of jtrees
 2. If cluster C_i has received the last missing message from neighbour C_j , C_i sends messages m_{ik} to all other neighbours C_k
 - In a parallel execution, it may happen that a central cluster receives all messages at once; this cluster can then send messages to all neighbours
- Induces a 2-phase message pass (to centre and back)

C_1 : Cond. 1 ✓
→ sends m_{12}

C_2 : Cond. 2 ✓
→ sends m_{21}, m_{23}

C_3 : Cond. 1 ✓
→ sends m_{32}



Message Passing

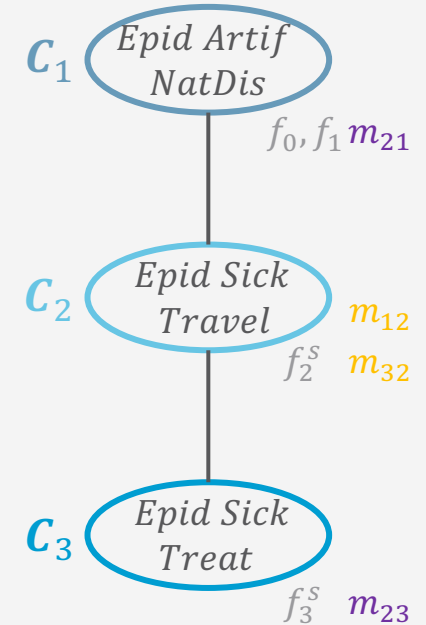
- Computing a message m_{ij} from cluster $\mathcal{C}_i$ to neighbour $\mathcal{C}_j$
 - Eliminate all non-separator variables in $\mathcal{C}_i$, from the union of the local models and the messages from the other neighbours $\mathcal{C}_{k \neq j}$
 - Call VE without normalisation at the end:

$$m_{ij} \leftarrow \text{VE-JT} \left(F_i \cup \bigcup_{\mathcal{C}_k \in \text{Nb}(\mathcal{C}_i), k \neq j} m_{ki}, \mathcal{S}_{ij}, \emptyset, . \right)$$

$\mathcal{C}_1$: Cond. 1 ✓
→ sends m_{12}

$\mathcal{C}_2$: Cond. 2 ✓
→ sends m_{21}, m_{23}

$\mathcal{C}_3$: Cond. 1 ✓
→ sends m_{32}



- Elimination order $\mathcal{U}$ for $\mathcal{C}_i \setminus \mathcal{S}_{ij}$ or heuristics h on last position
- Eliminates all random variables unknown to $\mathcal{C}_j$
- Messages from all other neighbours $\mathcal{C}_{k \neq j}$ make $\mathcal{C}_i$ independent from those parts of the model
- Union combines all information that is unknown to $\mathcal{C}_j$ and lie behind $\mathcal{S}_{ij}$ such that $\mathcal{C}_j$ is then independent of $\mathcal{C}_i$ given m_{ij}

Message Passing – Example

- $m_{12} \rightarrow \text{VE}(\{f_0, f_1\}, \{Epid\}, \emptyset, (Artif, NatDis))$

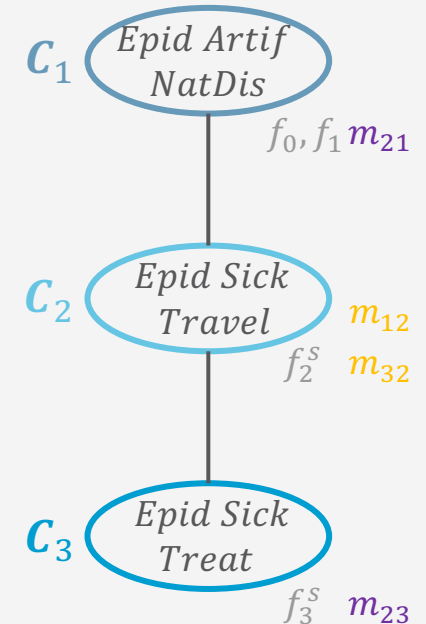
<i>Epid</i>	ϕ_0
false	50
true	1

<i>Epid</i>	<i>NatDis</i>	<i>Artif</i>	ϕ_1		<i>Epid</i>	<i>NatDis</i>	ϕ'_1		<i>Epid</i>	ϕ''_1
false	false	false	12	+	false	false	14	+	false	18
false	false	true	2							
false	true	false	3	+	false	true	4	+		
false	true	true	1							
true	false	false	7	+	true	false	11	+	true	17
true	false	true	4							
true	true	false	5	+	true	true	6	+		
true	true	true	1							

C_1 : Cond. 1 ✓
→ sends m_{12}

C_2 : Cond. 2 ✓
→ sends m_{21}, m_{23}

C_3 : Cond. 1 ✓
→ sends m_{32}



Message Passing – Example

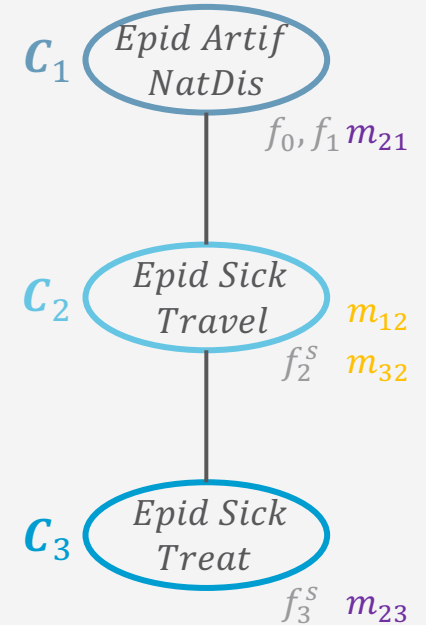
- $m_{12} \rightarrow \text{VE}(\{f_0, f_1\}, \{Epid\}, \emptyset, (Artif, NatDis))$
 - $m_{12} = \{f_0, f_1''\}$
- $m_{32} \rightarrow \text{VE}(\{f_3^s\}, \{Epid, Sick\}, \emptyset, (Treat))$
 - $m_{32} = \{f_3^{s'}\}$
- $m_{21} \rightarrow \text{VE}(\{f_2^s, f_3^{s'}\}, \{Epid\}, \emptyset, (Travel, Sick))$
 - $m_{21} = \{f_2^{s'}, f_3^{s'}\}$
- $m_{23} \rightarrow \text{VE}(\{f_2^s, f_0, f_1''\}, \{Epid, Sick\}, \emptyset, (Travel))$
 - $m_{23} = \{f_2^{s'}, f_0, f_1''\}$

Sick set with evidence and therefore eliminated through absorption. Summing out *Sick* does not have any effect here.

C_1 : Cond. 1 ✓
→ sends m_{12}

C_2 : Cond. 2 ✓
→ sends m_{21}, m_{23}

C_3 : Cond. 1 ✓
→ sends m_{32}



Algorithm for Message Passing

- Given a heuristics h
- While there is a cluster $\mathcal{C}_i$, which has not yet sent messages to all neighbours, i.e., $\sum_{j \in \text{Nb}(\mathcal{C}_i)} 1 \mid m_{ij} \text{ sent} \leq |\text{Nb}(\mathcal{C}_i)|$,
 - If Condition 1 with neighbour $\mathcal{C}_j$ holds
 - Compute message m_{ij} : $m_{ij} \leftarrow \text{VE-JT}(F_i \cup_{\mathcal{C}_k \in \text{Nb}(\mathcal{C}_i)} m_{ki}, \mathcal{S}_{ij}, h)$
 - Send m_{ij} to $\mathcal{C}_j$
 - If Condition 2 holds
 - For all neighbours $\mathcal{C}_j$ that did not yet receive a message
 - Compute m_{ij} : $m_{ij} \leftarrow \text{VE-JT}(F_i \cup_{\mathcal{C}_k \in \text{Nb}(\mathcal{C}_i), k \neq j} m_{ki}, \mathcal{S}_{ij}, h)$
 - Send m_{ij} an $\mathcal{C}_j$

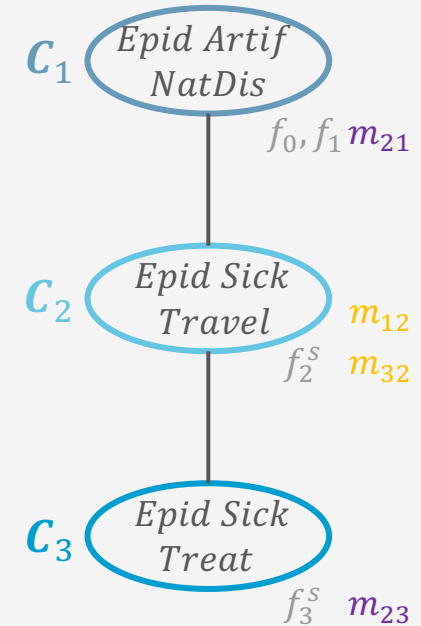
JT Step 3

Query Answering

- Result after Step 3: Clusters are independent given messages
- Simple case: Query variables Q appear together in a cluster
 - For each query Q
 - Find a cluster C_i with $Q \subseteq C_i$
 - Eliminate all non-query variables in local model and received messages:

$$\text{VE} \left(F_i \cup \bigcup_{C_j \in \text{Nb}(C_i)} m_{ji}, Q, \emptyset, . \right)$$

- Example:
 - $Q = \{Travel\}$ appears in $C_2 \rightarrow \text{VE}(\{f_2^S, f_0, f_1'', f_3^{S'}\}, \{Travel\}, \emptyset, .)$
 - VE multiplies all factors and sums out *Epid*
 - $Q = \{Epid\}$: All clusters possible
 - Optimisation criterion: Smallest cluster, fewest factors in local model, first cluster found



Query Answering

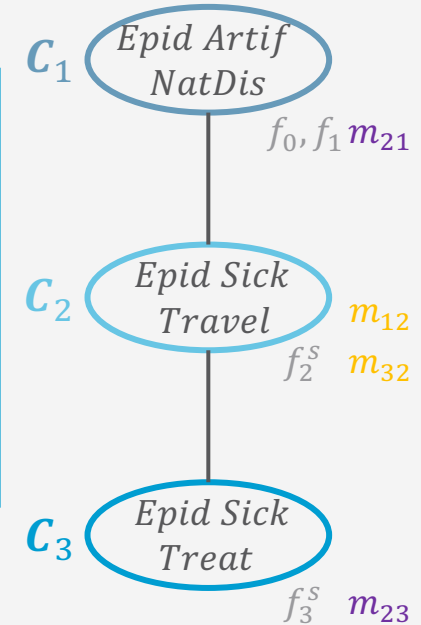
- Result after Step 3: Clusters are independent given messages
- General case:

- For each query Q
 - Find a **sub-jtree** J' with $Q \subseteq \text{rv}(J')$
 - Optimisation criterion: Number of clusters, number of random variables, first cluster found
 - Eliminate all non-query variables in local models and the messages from border of J' :

$$\text{VE} \left(\bigcup_{C_i \in J'} F_i \cup \bigcup_{C_j \in \text{Nb}(C_i), C_j \notin J'} m_{ji}, Q, \emptyset, . \right)$$

JT Step 4

- Example:
 - $Q = \{Travel, Treat\}$ appears in $C_2, C_3 \rightarrow \text{VE}(\{f_2^s, f_3^s, f_0, f_1''\}, \{Travel, Treat\}, \emptyset, .)$
 - VE multiplies all factors and sums out *Epid*

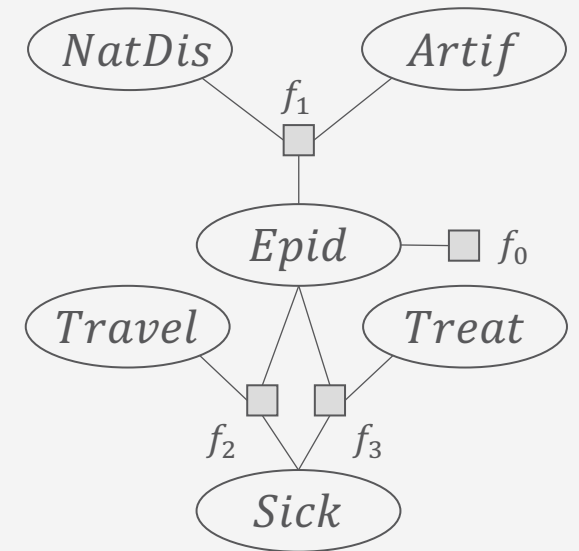
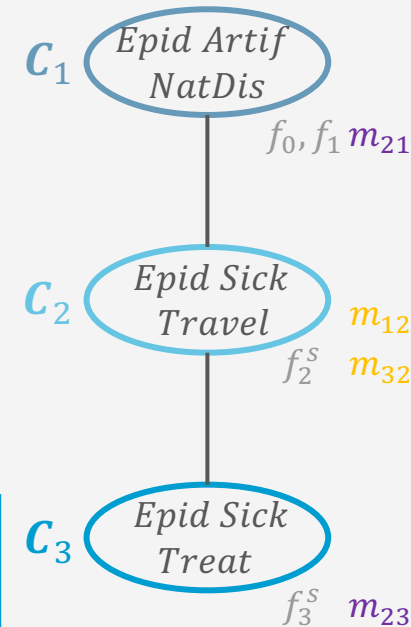


Junction Tree Algorithm (JT)

- Input
 - Model F
 - Evidence e
 - Query variables $\{Q_i\}_{i=1}^m$
 - Queries asked for $F: P(Q_i \mid e)$ for $i \in \{1, \dots, m\}$
- JT consists of four steps:

1. Build a jtree J for model F
2. Enter evidence e in J
3. Pass messages in J
4. Answer queries $\{Q_i\}_{i=1}^m$
 - Description of steps in blue boxes on previous slides

JT

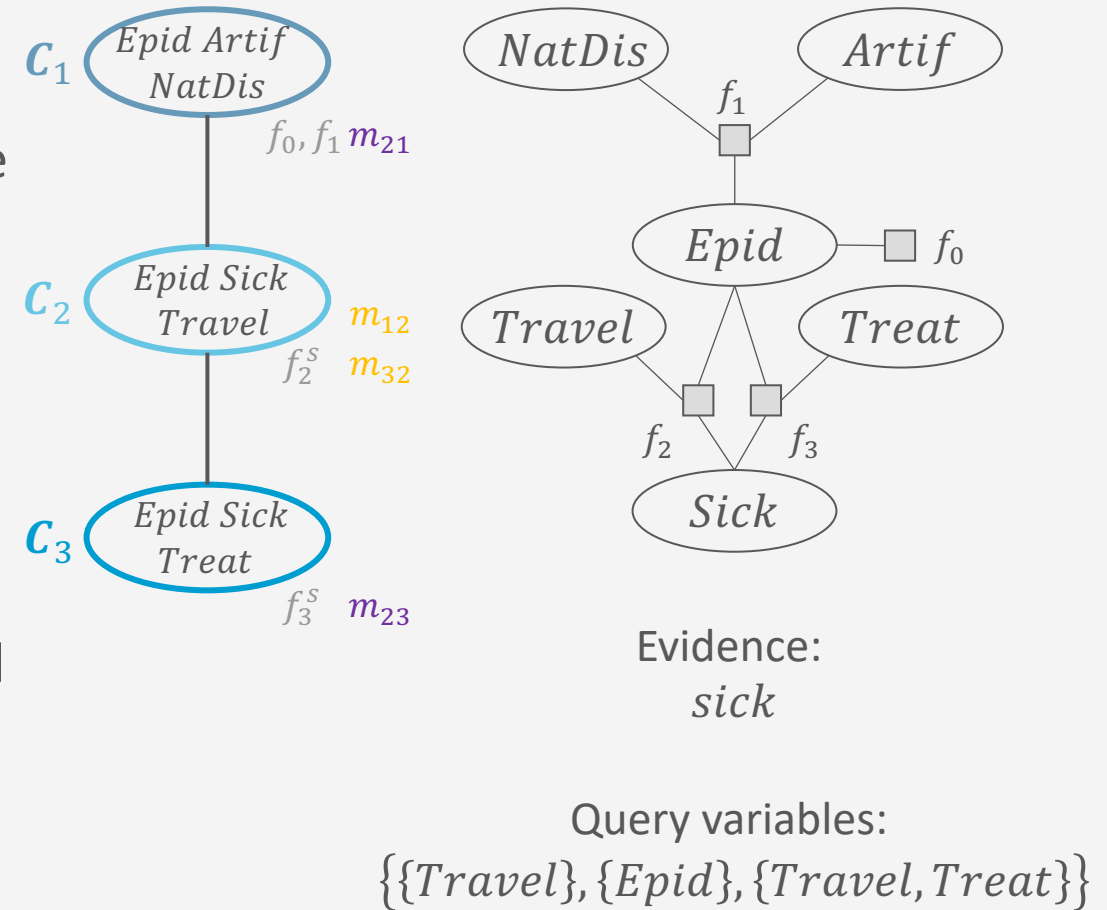


Evidence:
sick

Query variables:
 $\{\{Travel\}, \{Epid\}, \{Travel, Treat\}\}$

About Efficiency

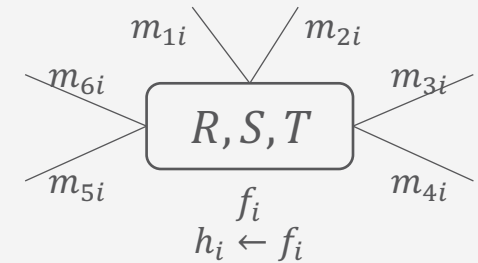
- Elimination order
 - Find one at the beginning when constructing dtree
 - For messages and queries in JT
 - Smaller search space as only the cluster needs to be considered
 - VE has to search for each query over all variables
- Could reduce local models to single factor using multiplication
 - + Possibly reduced number of multiplications overall
 - Possibly hides independences or enlarges factors



About Efficiency: Message Passing

- Message passing strategy so far: **Shafer-Shenoy architecture** [Shafer and Shenoy, 1989]
 - Downside: possibly many repeated multiplications of messages
 - In jtrees with high degree (many neighbours)
 - Even if local model reduced to one factor
 - Example: For each outgoing message, only one incoming message changes
- Alternative: **Hugin architecture** [Jensen et al., 1989]
 - Hugin factor $h_i = \phi_i(\mathbf{C}_i)$ for each cluster $\mathbf{C}_i$ as reduced local model
 - Multiply (and store) incoming messages m_{ji} : $h_i \leftarrow h_i \cdot m_{ji}$
 - When calculating message m_{ij} back: $\text{VE-JT}(h_i / m_{ji}, S_{ij}, \emptyset, .)$
 - Divide h_i by m_{ji} to not send back information that already com from there $\rightarrow$ single division instead of multiple multiplications

$$\begin{aligned}
 m_{i1} &\leftarrow \text{VE-JT}(\{f_i, m_{2i}, m_{3i}, m_{4i}, m_{5i}, m_{6i}\}, \dots) \\
 m_{i2} &\leftarrow \text{VE-JT}(\{f_i, m_{1i}, m_{3i}, m_{4i}, m_{5i}, m_{6i}\}, \dots) \\
 m_{i3} &\leftarrow \text{VE-JT}(\{f_i, m_{1i}, m_{2i}, m_{4i}, m_{5i}, m_{6i}\}, \dots) \\
 m_{i4} &\leftarrow \text{VE-JT}(\{f_i, m_{1i}, m_{2i}, m_{3i}, m_{5i}, m_{6i}\}, \dots) \\
 m_{i5} &\leftarrow \text{VE-JT}(\{f_i, m_{1i}, m_{2i}, m_{3i}, m_{4i}, m_{6i}\}, \dots) \\
 m_{i6} &\leftarrow \text{VE-JT}(\{f_i, m_{1i}, m_{2i}, m_{3i}, m_{4i}, m_{5i}\}, \dots)
 \end{aligned}$$



$$\begin{array}{ll}
 h_i \leftarrow h_i \cdot m_{1i} & h_i / m_{1i} \rightarrow \text{VE-JT} \\
 h_i \leftarrow h_i \cdot m_{2i} & h_i / m_{2i} \rightarrow \text{VE-JT} \\
 h_i \leftarrow h_i \cdot m_{3i} & h_i / m_{3i} \rightarrow \text{VE-JT} \\
 h_i \leftarrow h_i \cdot m_{4i} & h_i / m_{4i} \rightarrow \text{VE-JT} \\
 h_i \leftarrow h_i \cdot m_{5i} & h_i / m_{5i} \rightarrow \text{VE-JT} \\
 h_i \leftarrow h_i \cdot m_{6i} & h_i / m_{6i} \rightarrow \text{VE-JT}
 \end{array}$$

JT Runtime Complexity

- JT runtime complexity follows VE runtime complexity
 - Remember: VE complexity with a dtree T with n_T inner nodes
$$O(n_T \cdot r^w)$$
 - $r = \max_{R \in \mathcal{R}} |\text{ran}(R)|$ largest range size
 - $w = \max_{T_i \in \text{Desc}(T)} |\text{cluster}(T_i)|$ treewidth, i.e., largest cluster
 - Jtree J constructed from dtree T using clusters $\rightarrow$ treewidth in JT and VE equivalent
 - $w = \max_{T_i \in \text{Desc}(T)} |\text{cluster}(T_i)| = \max_{C_i \in J} |C_i|$
 - Due to J being minimal, different number of nodes possible
- Computations in JT steps are calls to VE
 - VE operations on a factor bounded by $O(r^w)$
- To simplify, consider jtree with a single factor at each cluster and singleton queries $\{Q_i\}_{i=1}^m$

JT Step-wise Complexity

- Jtree construction: *depends on heuristics*
- Handling evidence: $O(e \cdot n_J \cdot r^w)$
- Passing messages: $O(n_J \cdot r^w)$
- Answering queries: $O(m \cdot r^w)$
- Together: $O((e \cdot n_J + m) \cdot r^w)$
 - $(e \cdot n_J \cdot r^w) + (n_J \cdot r^w) + (m \cdot r^w) = (e \cdot n_J + n_J + m) \cdot r^w = ((e - 1) \cdot n_J + m) \cdot r^w$
- As VE complexity of $O(n_T \cdot r^w)$ without evidence, comparable JT complexity is:
$$O((n_J + m) \cdot r^w)$$

JT Runtime Complexity

- Runtime complexity without evidence:

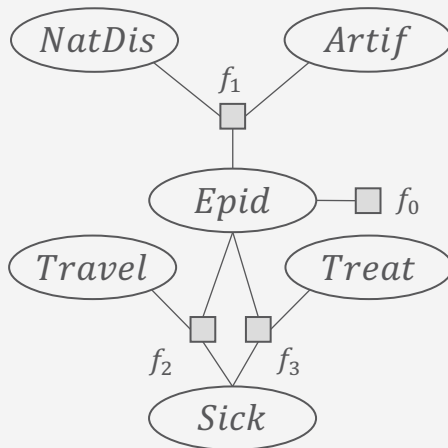
$$O\left((n_J + m) \cdot r^w\right)$$

- VE complexity $O(n_T \cdot r^w)$ almost the complexity of message passing $O(n_J \cdot r^w)$
 - In practice, twice the computations as two messages for each edge
 - Overhead needs two to three queries to pay off
- VE complexity for m queries: $O(m \cdot n_T \cdot r^w)$
 - If asking a query for each random variable, $m = |\mathbf{R}| = n = n_T$: $O(n^2 \cdot r^w)$ vs. $O(n \cdot r^w)$
 - As $n \geq n_J$: $O((n_J + n) \cdot r^w) \leq O((n + n) \cdot r^w) = O(2n \cdot r^w) \rightarrow O(n \cdot r^w)$
- If abstracting away preprocessing steps, complexity for each query:
 $O(n_T \cdot r^w)$ vs. $O(r^w)$

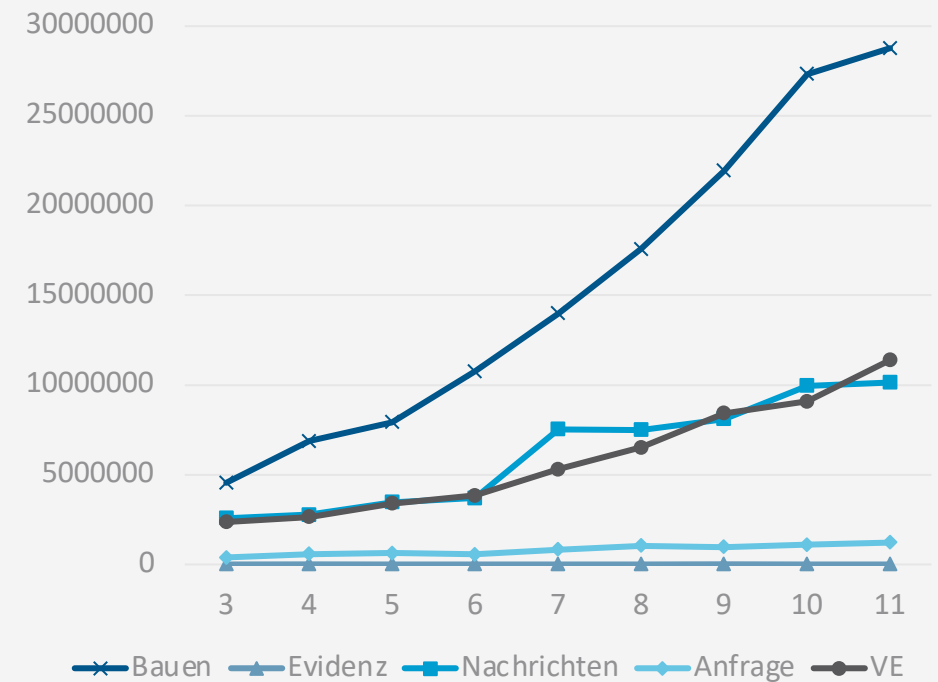
JT Implementation: Step-wise Runtimes (without Evidence)

- Runtime complexity of JT:

$$O\left((n_J + m) \cdot r^w\right)$$
- Behaviour with increasing
 - w : Add more random variables to factors



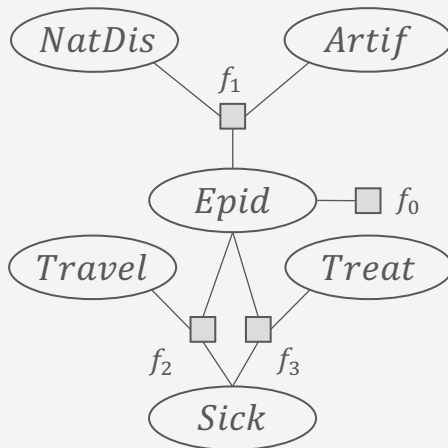
- Runtime:
 - Construction takes up most time



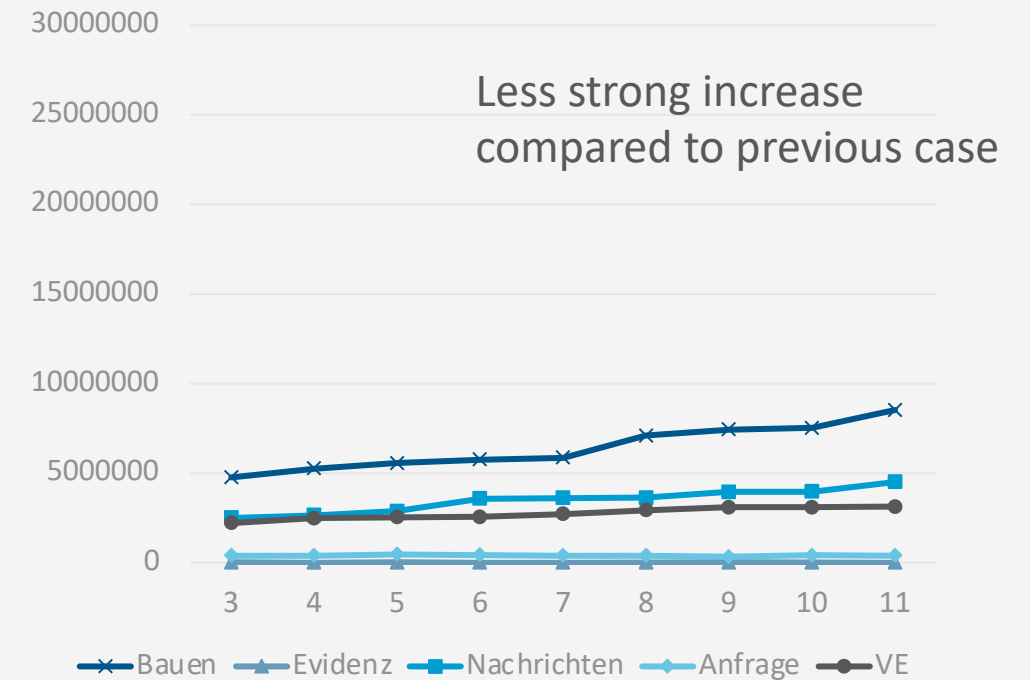
JT Implementation: Step-wise Runtimes (without Evidence)

- Runtime complexity of JT:

$$O\left((n_J + m) \cdot r^w\right)$$
- Behaviour with increasing
 - n_J : Add more factors to *Epid* with two new random variables → form own cluster



- Runtime:
 - Construction takes up most time

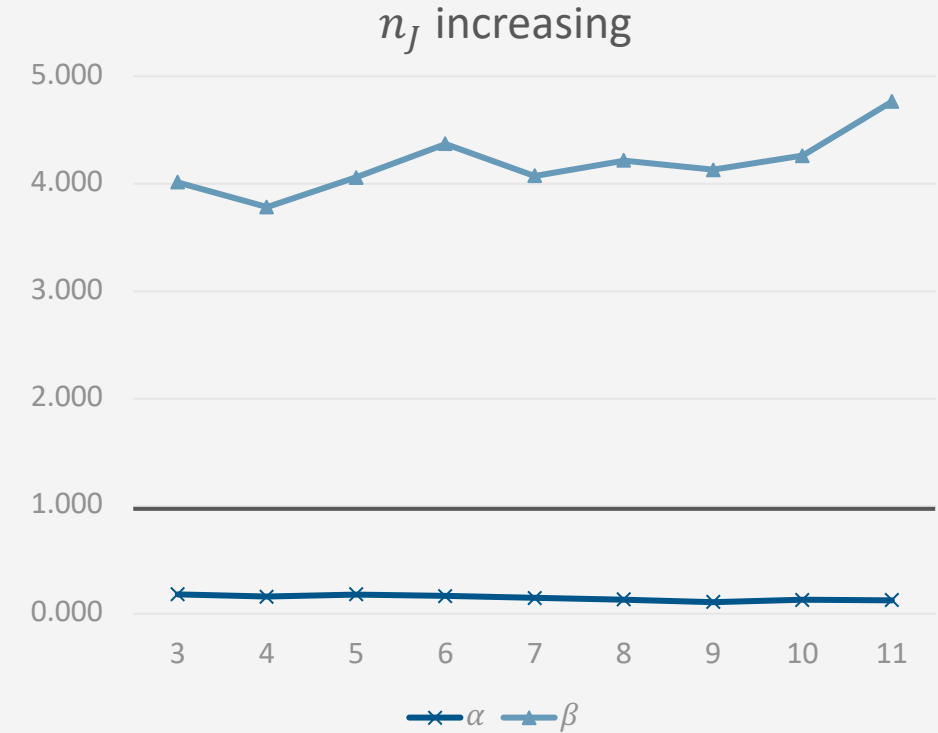
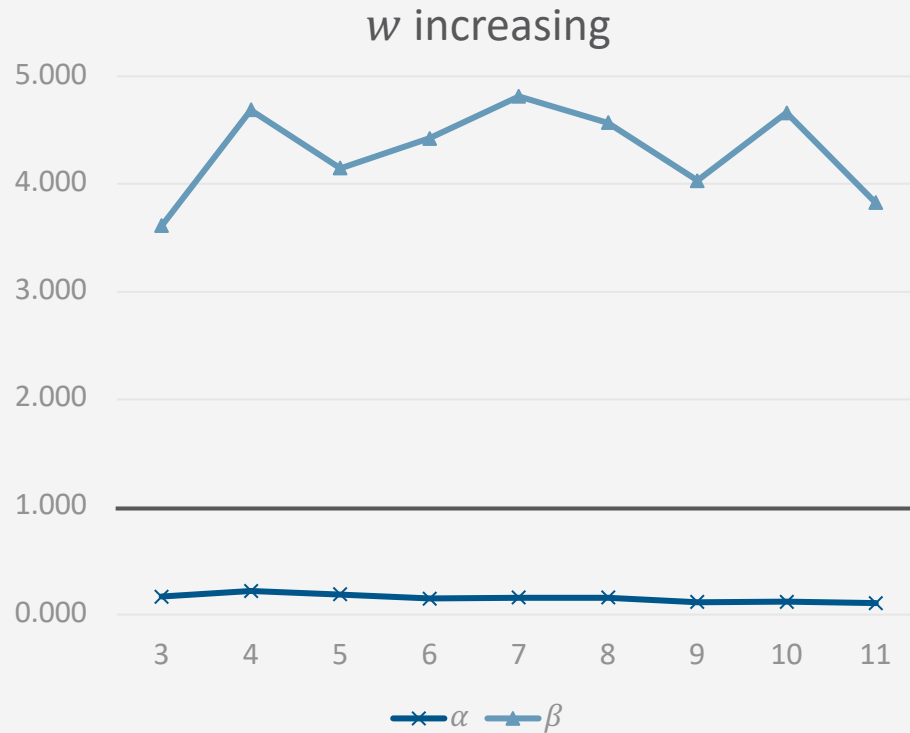


Comparison Implementation Runtime

- Trade-off for multi-query algorithms
 - Overhead, which has to be offset (model *compiled* into a helper structure)
vs.
 - Shorter runtime during query answering for individual queries
- Comparison statistics:
 - $t_{q,cpl}$ runtime for answering a query with a compilation algorithm
 - $t_{q,uncpl}$ runtime for answering a query without compilation
 - $t_{c,cpl}$ runtime for compilation
 - Ratio of runtime for query answering: $\alpha = \frac{t_{q,cpl}}{t_{q,uncpl}}$
 - Number of queries until overhead is offset: $\beta = \frac{t_{c,cpl}}{t_{q,uncpl} - t_{q,cpl}}$
 - Only reasonable if $\alpha > 1$

Comparison Implementation Runtime

- Comparison statistics: $\alpha = \frac{t_{q,cpl}}{t_{q,uncpl}}$ and $\beta = \frac{t_{c,cpl}}{t_{q,uncpl} - t_{q,cpl}}$



Interim Summary

- Jtree as helper structure
 - Clusters of random variables with separators between them
- JT steps
 - Construct jtree: Use dtree clusters
 - Enter evidence: Absorb in each cluster
 - Pass messages: two phases to make clusters independent, calculate message using VE based on local model and messages from other neighbours summing out non-separator variables
 - Answer queries: Find cluster or subtree over query variable(s) and use local model(s) and outside messages as a basis for VE
- Complexity: Message passing complexity corresponds to VE complexity
JT query answering complexity independent of number of nodes

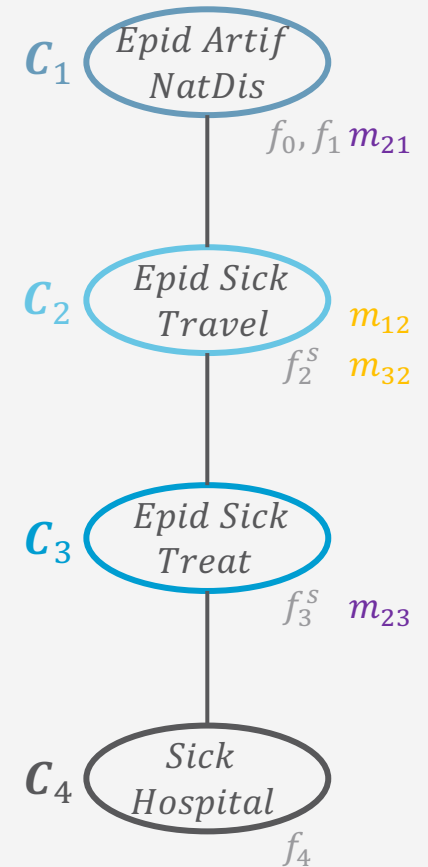
Beyond Standard Query Answering

Adaptive inference

Most probable assignment queries

Beyond Standard Query Answering

- **Adaptive** inference: Answer queries as fast as possible again after
 - Changes in query variables → Reuse jtree as is (including current local models and messages)
 - Standard use case of JT
 - Changes in evidence → Reset local models, enter evidence, send messages
 - Re-use jtree structure
 - Only have to send a message if evidence changed or an incoming message changed
 - Changes in model → Adapt jtree, resend messages
 - Re-use as much as possible of jtree structure
 - E.g., changes in potentials do not necessitate a new jtree
 - But if changes too dramatic or too many changes, have to rebuild from scratch
 - Send messages in new parts and from new parts to outer parts



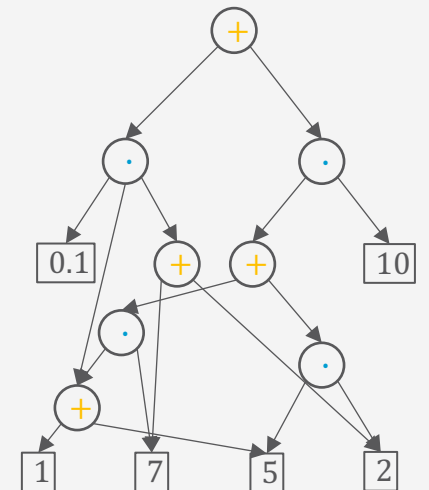
Beyond Standard Query Answering

- Most probable explanation (MPE)

- Compute most probable state of the remaining variables $\mathbf{R}$ given evidence $\mathbf{e}$
- Replace sum-out operations with max-out operations:

$$\text{MPE}_F(\mathbf{e}) = \arg \max_{\mathbf{r} \in \text{ran}(\mathbf{R})} P(\mathbf{r} \mid \mathbf{e}) = \arg \max_{\mathbf{r} \in \text{ran}(\mathbf{R})} P(\mathbf{r}, \mathbf{e})$$

- Like sum-out operations, move max-out operation into the product of factors
- In implementation, have to store assignments
- Same idea applies to knowledge compilation
 - Replace $+$ with max in inner disjunction nodes
 - Read out assignment from top to bottom by following edges of maximum value

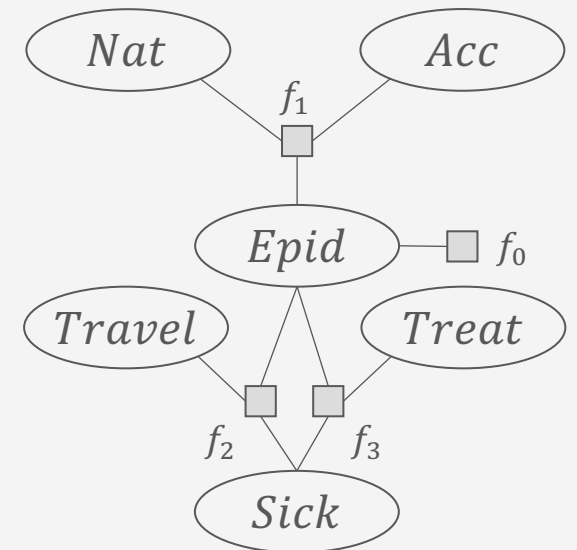


Beyond Standard Query Answering

- Generalisation: **Maximum-a-posteriori Assignment (MAP)**
 - Compute most probable state of a subset $\mathbf{U}$ of the remaining variables $\mathbf{R}$ given evidence $\mathbf{e}$
 - Sum-out the non-query variables $\mathbf{V} = \mathbf{R} \setminus \mathbf{U}$, max-out the query variables $\mathbf{U}$

$$\text{MAP}_F(\mathbf{U} \mid \mathbf{e}) = \arg \max_{\mathbf{u} \in \text{ran}(\mathbf{U})} \sum_{\mathbf{v} \in \text{ran}(\mathbf{V})} P(\mathbf{u}, \mathbf{v} \mid \mathbf{e}) = \arg \max_{\mathbf{u} \in \text{ran}(\mathbf{U})} \sum_{\mathbf{v} \in \text{ran}(\mathbf{V})} P(\mathbf{u}, \mathbf{v}, \mathbf{e})$$

- In standard query answering problem: $\mathbf{U} = \emptyset$ ($\mathbf{V} = \mathbf{R}$)
- In MPE: $\mathbf{U} = \mathbf{R}$ ($\mathbf{V} = \emptyset$)
- MAP harder problem as sum and max are not commutative
 - Can mess with elimination order
 - Example: $\text{MAP}_F(\text{Travel}, \text{Sick}, \text{Treat}, \text{Acc}, \text{Nat}) \rightarrow$ Requires summing out *Epid*, which requires multiplying all factors
 - Bounded MAP query: If $\mathbf{U}$ covers separators connected in a jtree, then MAP query does not increase width



Outline: 3. Inference

a. Exact Inference

i. (Weighted) Model Counting

- Model counting
- Weighted model counting, knowledge compilation (KC)

ii. Variable Elimination and the Like

- Variable elimination (VE)
- Junction tree algorithm (JT)

iii. Lifted Inference

- First-order KC, Lifted VE, lifted JT

b. Approximate Inference

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling

Appendix

Formal definitions of absorption, multiplication, summing out, as well as VE

Full VE example calculations

Example derivation of a junction tree, VE for message passing in JT

Absorption: Formal Definition

- Operator: **ABSORB**
 - Inputs:
 - Factor $f = \phi(R_1, \dots, R_n) \in F$
 - Variable $R \in \{R_1, \dots, R_n\}$ at position i
 - Factor $f^r = \phi(R)$ with mappings $r \mapsto 1$ and $\forall r' \neq r \in \text{ran}(R) : r' \mapsto 0$ for observation $R = r$,
 - Precondition: *none*
 - Output: Factor $\phi'(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$
 - For all possible valuations $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n$ of $R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n$
 - I.e., $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n \in \text{ran}(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$, with

$$\phi'(r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n) = \phi(r_1, \dots, r_{i-1}, r, r_{i+1}, \dots, r_n)$$

- Postcondition: $F \cup \{f^r\} \sim F \setminus \{f\} \cup \{f^r, \text{ABSORB}(f, R, f^r)\}$

Factor Multiplication: Formal Definition

- Operator: **MULTIPLY**
 - Inputs:
 - Factor $f_1 = \phi_1(R_1, \dots, R_n) \in F$
 - Factor $f_2 = \phi_2(S_1, \dots, S_m) \in F$
 - Precondition: *none*
 - Output: Factor $\phi(T_1, \dots, T_k)$
 - $\{(T_1, \dots, T_k)\} = \{(R_1, \dots, R_n)\} \bowtie \{(S_1, \dots, S_m)\}$ (ordered union)
 - For all possible valuations $t_1, \dots, t_k$ of $T_1, \dots, T_k$, i.e., $t_1, \dots, t_k \in \text{ran}(T_1, \dots, T_k)$, with
 - $r_1, \dots, r_n = \pi_{R_1, \dots, R_n}(t_1, \dots, t_k)$ and $s_1, \dots, s_m = \pi_{S_1, \dots, S_m}(t_1, \dots, t_k)$ (select corresponding values from $t_1, \dots, t_k$)

$$\phi(t_1, \dots, t_k) = \phi_1(r_1, \dots, r_n) \cdot \phi_2(s_1, \dots, s_m)$$

- Postcondition: $F \sim F \setminus \{f_1, f_2\} \cup \text{MULTIPLY}(f_1, f_2)$

Summing out Variables: Formal Definition

- Operator: SUM-OUT
 - Inputs:
 - Factor $f = \phi(R_1, \dots, R_n) \in F$
 - Variable $R \in \{R_1, \dots, R_n\}$ at position i to sum out
 - Precondition: $\forall f' \in F \setminus \{f\}: R \notin \text{rv}(f')$
 - Output: Factor $\phi'(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$
 - For each possible valuation $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n$ of $R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n$
 - i.e., $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n \in \text{ran}(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$

$$\phi'(r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n) = \sum_{r \in \text{ran}(R)} \phi(r_1, \dots, r_{i-1}, r, r_{i+1}, \dots, r_n)$$

- Postcondition: $\sum_{r \in \text{ran}(R)} P_F \equiv P_{F \setminus \{f\} \cup \text{SUM-OUT}(f, R)}$

VE Algorithm Using a Heuristics h

$\text{VE}(F, \mathcal{S}, \{\phi_t(T_t)\}_{t=1}^m, h)$

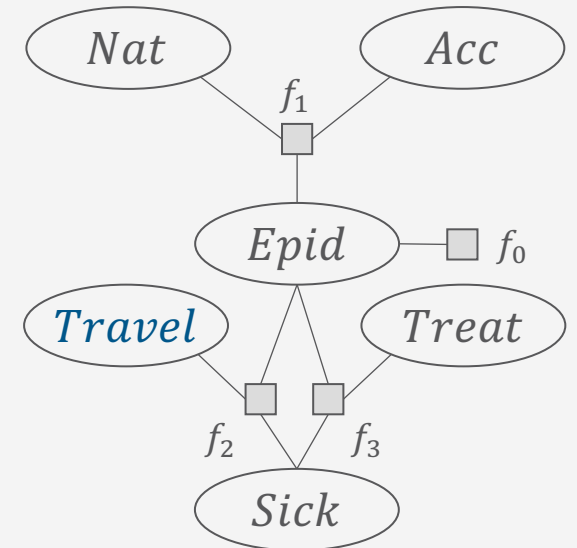
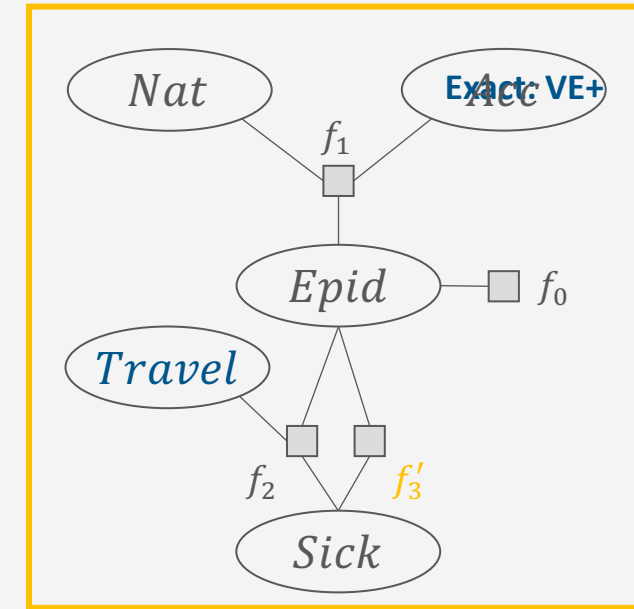
for $t = 1, \dots, m$ do	Handle evidence
while $\exists f \in F : T_t \in rv(f)$ do	$\triangleright$ Absorb $\phi_t(T_t)$ in F
$F \leftarrow F \setminus \{f\} \cup \{\text{ABSORB}(f, T, \phi_t(T))\}$	
while $rv(F) \setminus \mathcal{S} \neq \emptyset$ do	Eliminate non-query variables
$U \leftarrow \arg \min_R h(F)$	$\triangleright$ Choose next U to eliminate
while $\exists f_1, f_2 \in F : U \in rv(f_1) \wedge rv(f_2)$ do	
$F \leftarrow F \setminus \{f_1, f_2\} \cup \{\text{MULTIPLY}(f_1, f_2)\}$	$\triangleright$ Multiply f_1, f_2 in F
$F \leftarrow F \setminus \{f\} \cup \{\text{SUM-OUT}(f, U)\}$	$\triangleright$ Sum out U in F
while $\exists f_1, f_2 \in F$ do	$\triangleright$ Multiply f_1, f_2 in F , until $ F = 1$
$F \leftarrow F \setminus \{f_1, f_2\} \cup \{\text{MULTIPLY}(f_1, f_2)\}$	
Normalise the potentials in the one remaining $f \in F$	Normalise
return f	

$P(\text{Travel})$

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

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_2(\text{Travel}, e, s) \phi'_3(e, s)$$

<i>Epid</i>	<i>Sick</i>	<i>Treat</i>	ϕ_3		<i>Epid</i>	<i>Sick</i>	ϕ'_3
false	false	false	5	+	false	false	6
false	false	true	1				
false	true	false	3	+	false	true	5
false	true	true	2				
true	false	false	5	+	true	false	9
true	false	true	4				
true	true	false	1	+	true	true	8
true	true	true	7				

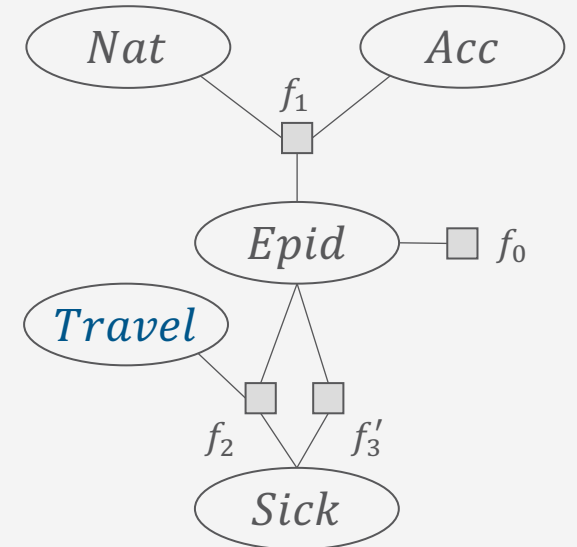
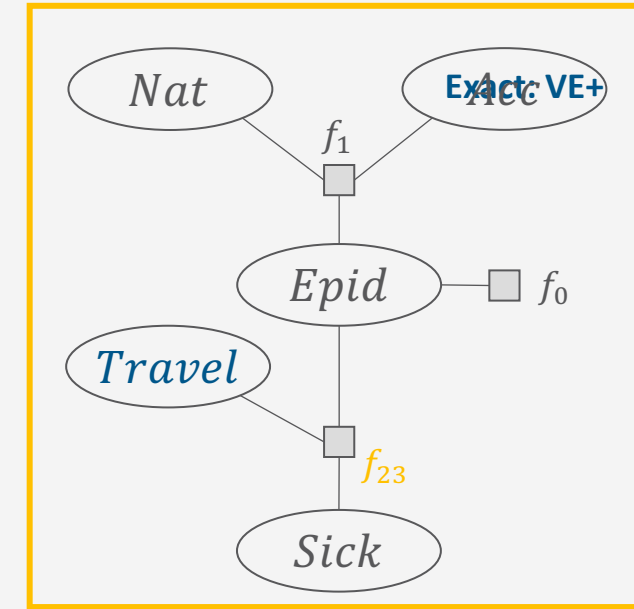


$P(Travel)$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \boxed{\phi_2(Travel, e, s) \phi'_3(e, s)}$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_{23}(Travel, e, s)$$

<i>Travel</i>	<i>Epid</i>	<i>Sick</i>	ϕ_2	<i>Epid</i>	<i>Sick</i>	ϕ'_3	<i>Travel</i>	<i>Epid</i>	<i>Sick</i>	ϕ_{23}
false	false	false	20	false	false	6	false	false	false	$20 \cdot 6 = 120$
false	false	true	24	false	true	5	false	false	true	$24 \cdot 5 = 120$
false	true	false	5	true	false	9	false	true	false	$5 \cdot 9 = 45$
false	true	true	6	true	true	8	false	true	true	$6 \cdot 8 = 48$
true	false	false	28				true	false	false	$28 \cdot 6 = 168$
true	false	true	8				true	false	true	$8 \cdot 5 = 40$
true	true	false	7				true	true	false	$7 \cdot 9 = 63$
true	true	true	2				true	true	true	$2 \cdot 8 = 16$

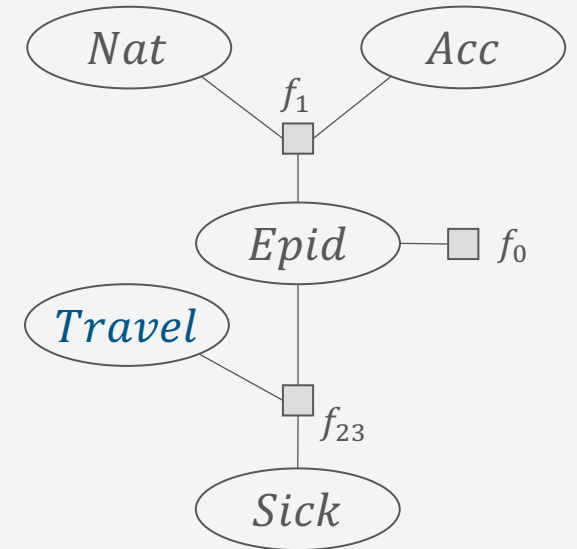
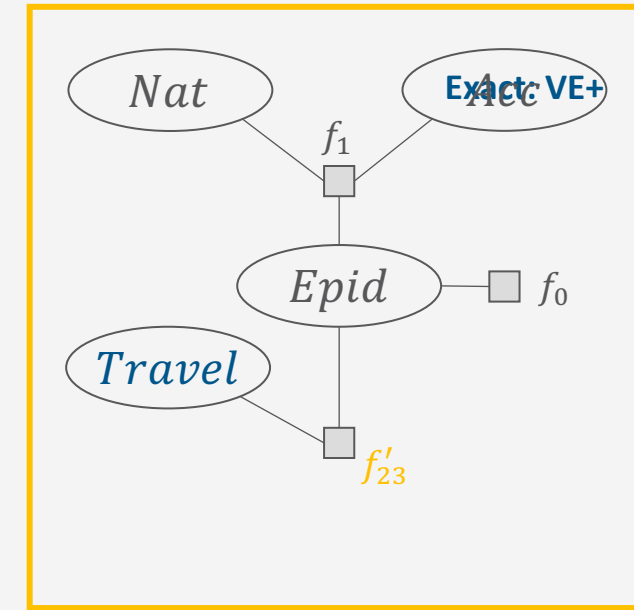


$P(\text{Travel})$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \sum_{s \in \text{Val}(S)} \phi_{23}(\text{Travel}, e, s)$$

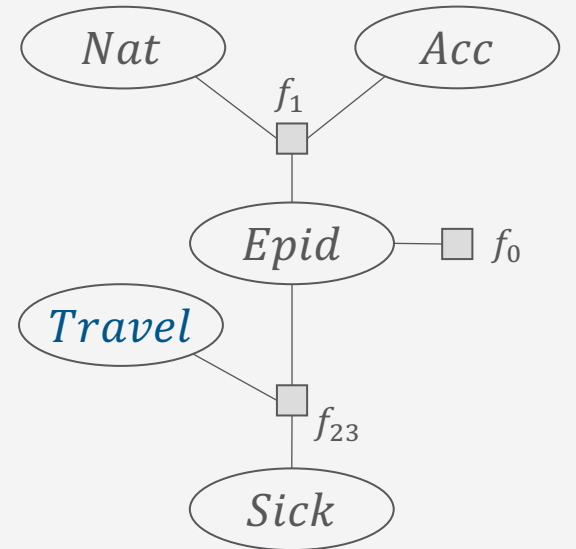
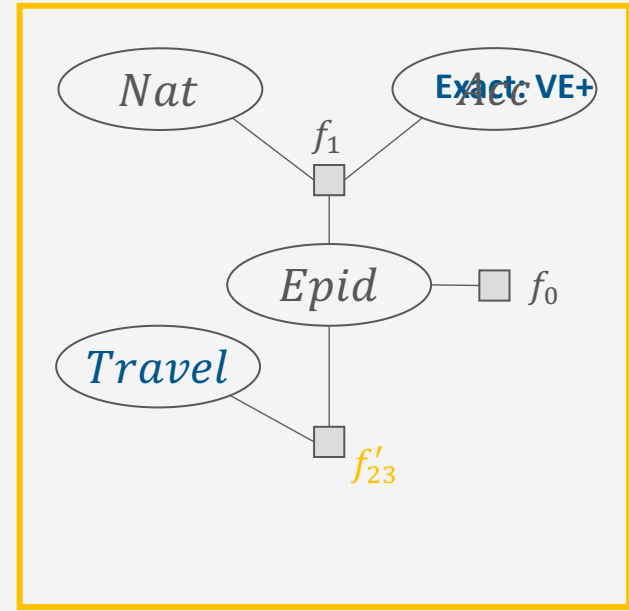
$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \phi'_{23}(\text{Travel}, e)$$

<i>Travel</i>	<i>Epid</i>	<i>Sick</i>	ϕ_{23}		<i>Travel</i>	<i>Epid</i>	ϕ'_{23}
false	false	false	120	+	false	false	240
false	false	true	120				
false	true	false	45	+	false	true	93
false	true	true	48				
true	false	false	168	+	true	false	208
true	false	true	40				
true	true	false	63	+	true	true	79
true	true	true	16				



$P(\textit{Travel})$

$$\begin{aligned}
 &\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a) \phi'_{23}(\textit{Travel}, e) \\
 &\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(\textit{Travel}, e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a)
 \end{aligned}$$

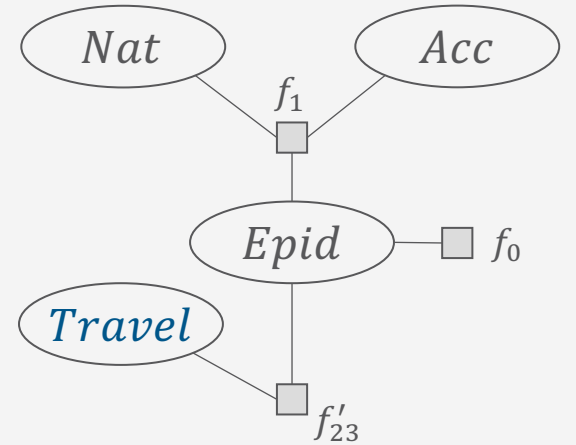
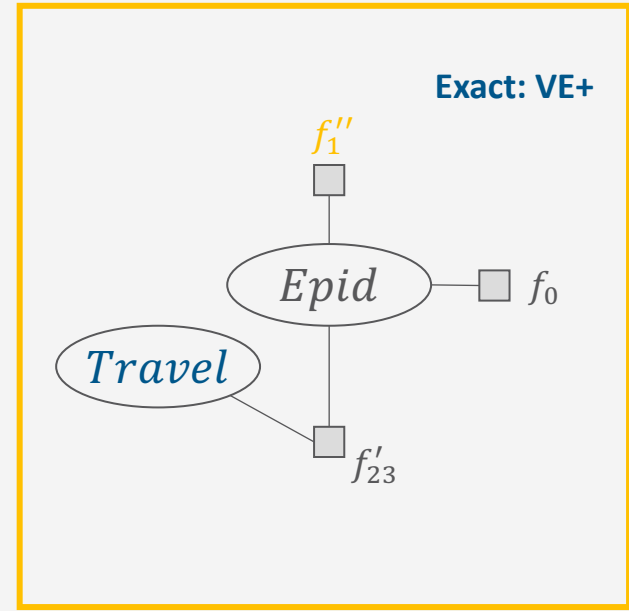


$P(Travel)$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(Travel, e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(Travel, e) \phi''_1(e)$$

<i>Epid</i>	<i>NatDis</i>	<i>Artif</i>	ϕ_1		<i>Epid</i>	<i>NatDis</i>	ϕ'_1		<i>Epid</i>	ϕ''_1
false	false	false	12	+	false	false	14	+	false	18
false	false	true	2							
false	true	false	3	+	false	true	4	+		
false	true	true	1							
true	false	false	7	+	true	false	11	+	true	17
true	false	true	4							
true	true	false	5	+	true	true	6	+		
true	true	true	1							



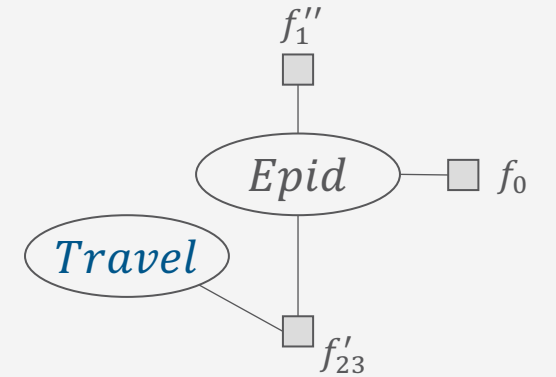
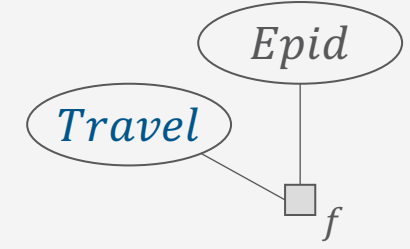
$P(Travel)$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi'_{23}(Travel, e) \phi''_1(e)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi(Travel, e)$$

<i>Travel</i>	<i>Epid</i>	ϕ'_{23}	<i>Epid</i>	ϕ''_1	<i>Epid</i>	ϕ_0	<i>Travel</i>	<i>Epid</i>	ϕ
false	false	240	false	18	false	50	false	false	$240 \cdot 18 \cdot 50 = 216,000$
false	true	93	true	17	true	1	false	true	$93 \cdot 17 \cdot 1 = 1,581$
true	false	208					true	false	$208 \cdot 18 \cdot 50 = 187,200$
true	true	79					true	true	$79 \cdot 17 \cdot 1 = 1,343$

Exact: VE+

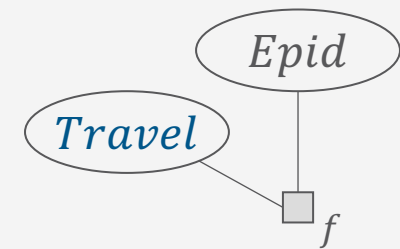
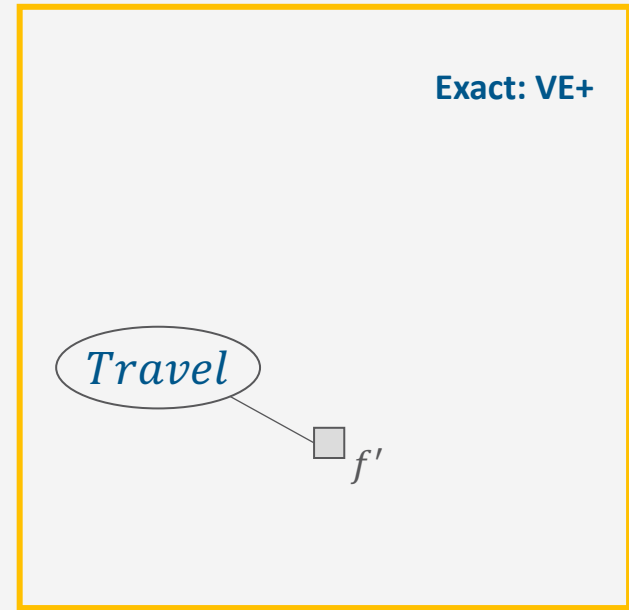


$P(Travel)$

$$\propto \sum_{e \in \text{Val}(E)} \phi(Travel, e)$$

$$\propto \phi'(Travel)$$

<i>Travel</i>	<i>Epid</i>	ϕ		<i>Travel</i>	ϕ'
<i>false</i>	<i>false</i>	216,000	+	<i>false</i>	217,581
<i>false</i>	<i>true</i>	1,581			
<i>true</i>	<i>false</i>	187,200	+	<i>true</i>	188,543
<i>true</i>	<i>true</i>	1,343			



$P(\text{Travel})$

$$\propto \phi'(\text{Travel})$$

$$= \phi^n(\text{Travel})$$

$$= P(\text{Travel})$$

<i>Travel</i>	ϕ		<i>Travel</i>	ϕ^n
<i>false</i>	217,581		<i>false</i>	$\frac{217,581}{217,581 + 188,543} = \frac{217,581}{406,124} = 0.54$
<i>true</i>	188,543		<i>true</i>	$\frac{188,543}{217,581 + 188,543} = \frac{188,543}{406,124} = 0.46$

Exact: VE+



VE with Evidence: Example

$$P(\text{Travel} \mid \text{sick})$$

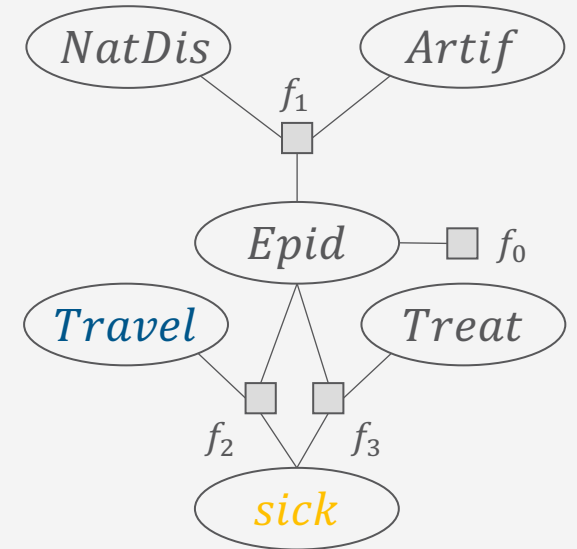
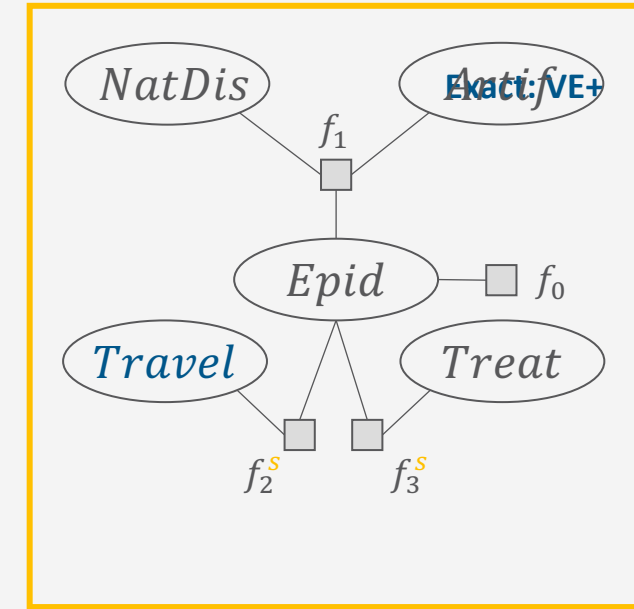
$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{t \in \text{Val}(T)} P_R(E = e, N = n, A = a, \text{sick}, \text{Travel}, T = t)$$

$$\propto \sum_{e \in \text{Val}(E)} \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \sum_{t \in \text{Val}(T)} \prod_{i=0}^3 \phi_i(R_i = r_i)$$

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

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

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



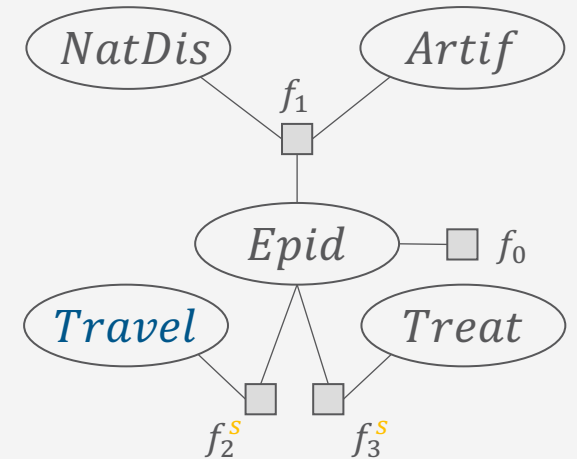
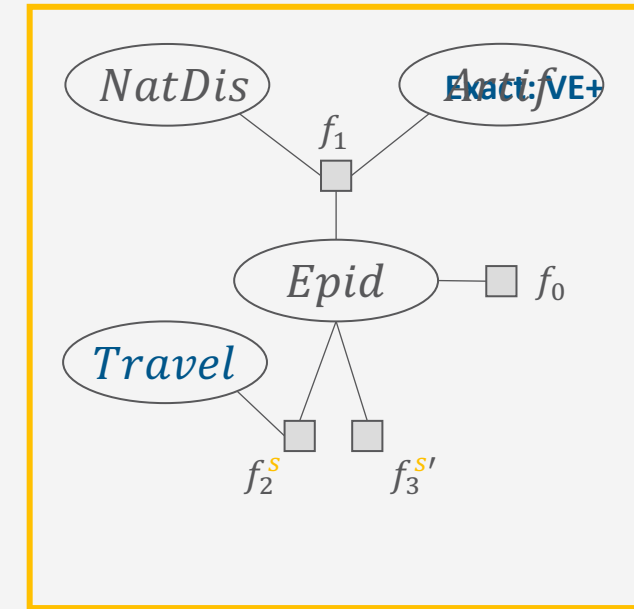
VE with Evidence: Example

$P(\text{Travel} \mid \text{Sick})$

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

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

<i>Epid</i>	<i>Treat</i>	ϕ_3^S		<i>Epid</i>	$\phi_3^{S'}$
false	false	3	+	false	5
false	true	2	+	true	8
true	false	1			
true	true	7			



VE with Evidence: Example

$$P(\text{Travel} \mid \text{sick})$$

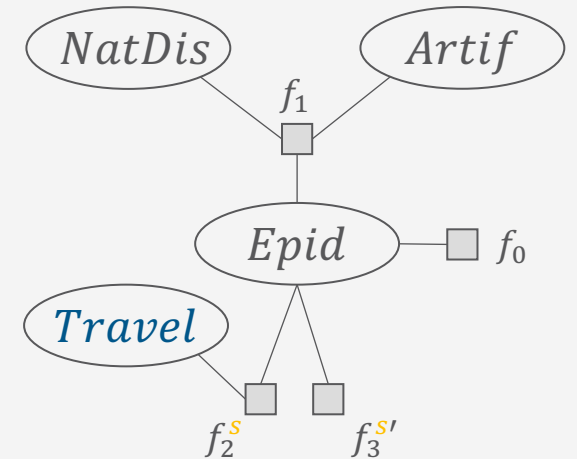
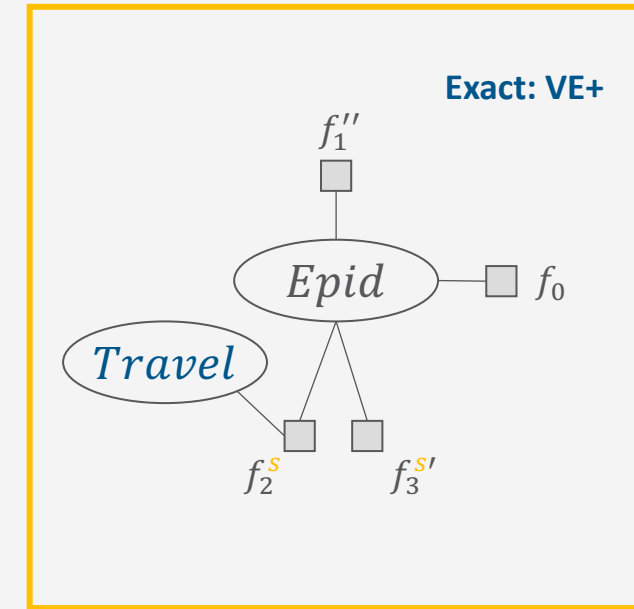
$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi_2^s(\text{Travel}, e) \phi_3^{s'}(e) \sum_{n \in \text{Val}(N)} \sum_{a \in \text{Val}(A)} \phi_1(e, n, a)$$

Wie vorher

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi_2^s(\text{Travel}, e) \phi_3^{s'}(e) \phi_1(e'')$$

<i>Epid</i>	<i>NatDis</i>	<i>Artif</i>	ϕ_1
false	false	false	12
false	false	true	2
false	true	false	3
false	true	true	1
true	false	false	7
true	false	true	4
true	true	false	5
true	true	true	1

<i>Epid</i>	ϕ_1''
false	18
true	17



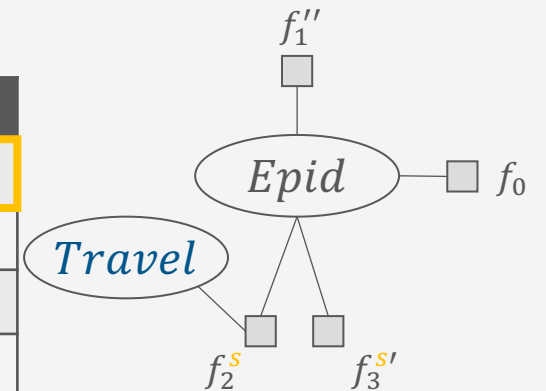
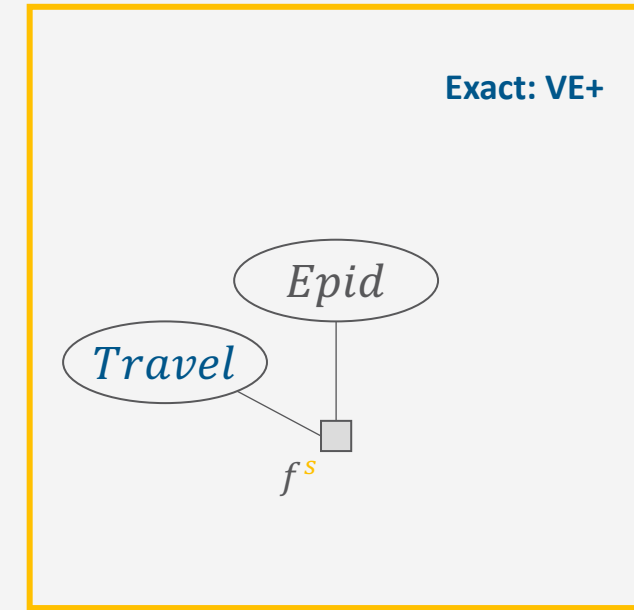
VE with Evidence: Example

$$P(\text{Travel} \mid \text{sick})$$

$$\propto \sum_{e \in \text{Val}(E)} \phi_0(e) \phi_2^s(\text{Travel}, e) \phi_3^{s'}(e) \phi_1''(e)$$

$$\propto \sum_{e \in \text{Val}(E)} \phi^s(\text{Travel}, e)$$

Travel	Epid	ϕ_2^s	Epid	ϕ_0	Epid	$\phi_3^{s'}$	Epid	ϕ_1''	Travel	Epid	ϕ
false	false	24	false	50	false	5	false	18	false	false	$24 \cdot 50 \cdot 5 \cdot 18 = 108.000$
false	true	6	true	1	true	8	true	17	false	true	$6 \cdot 1 \cdot 8 \cdot 17 = 816$
true	false	8							true	false	$8 \cdot 50 \cdot 5 \cdot 18 = 36.000$
true	true	2							true	true	$2 \cdot 1 \cdot 8 \cdot 17 = 272$

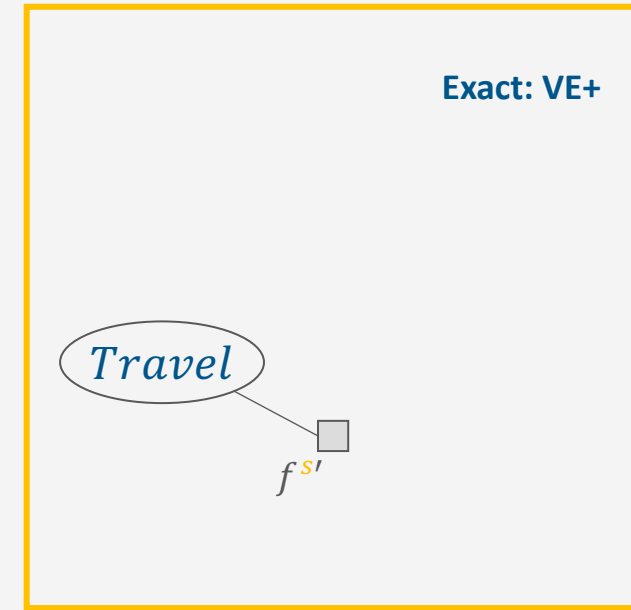
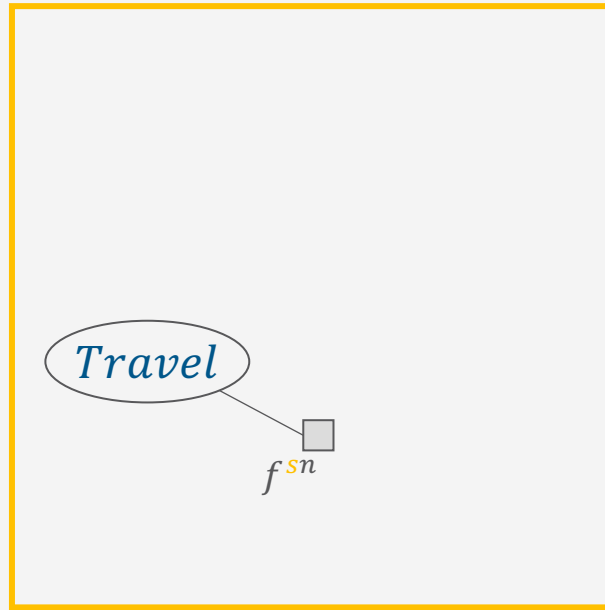


VE with Evidence: Example

$$P(\text{Travel} \mid \text{sick})$$

$$\propto \sum_{e \in \text{Val}(E)} \phi^s(\text{Travel}, e)$$

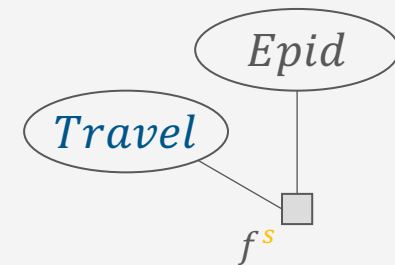
$$\begin{aligned}
 &\propto \phi^{s'}(\text{Travel}) \\
 &= \phi^{sn}(\text{Travel}) \\
 &= P(\text{Travel} \mid \text{sick})
 \end{aligned}$$



<i>Travel</i>	<i>Epid</i>	ϕ^s
false	false	108.000
false	true	816
true	false	36.000
true	true	272

<i>Travel</i>	$\phi^{s'}$
false	108.816
true	36.272

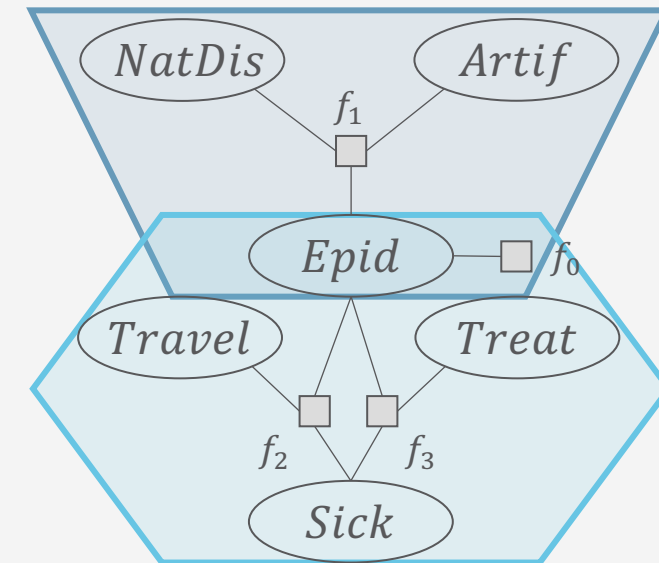
<i>Travel</i>	ϕ^{sn}
false	$\frac{108.816}{108.816 + 36.272} = \frac{108.816}{145.088} = 0.75$
true	$\frac{36.272}{108.816 + 36.272} = \frac{36.272}{145.088} = 0.25$



Example

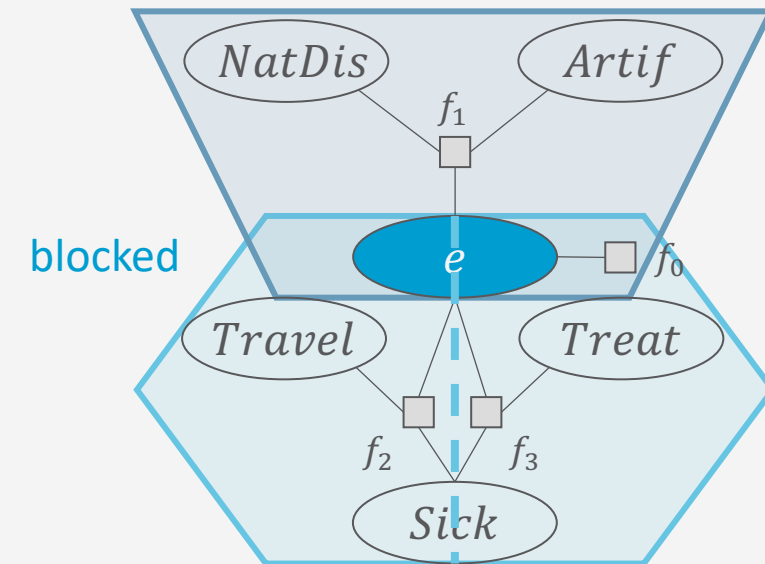
- Set of queries
 - $P(Epid), P(NatDis), P(Artif), P(Travel), P(Treat), P(Sick)$
 - For $P(NatDis), P(Artif)$, the *lower* part of the model ...
 - For $P(Travel), P(Treat), P(Sick)$, the *upper* part of the model ...
- ... would be summed out in the same way
- Would it not be nice if for the queries
 - $P(NatDis), P(Artif)$ only the *upper* part of the model ...
 - $P(Travel), P(Treat), P(Sick)$ only the *lower* part of the model ...
- ... would be relevant?
- And whatever else happens outside of those parts would be available beforehand?

$P(Epid)$ could be considered as part of the upper or lower part



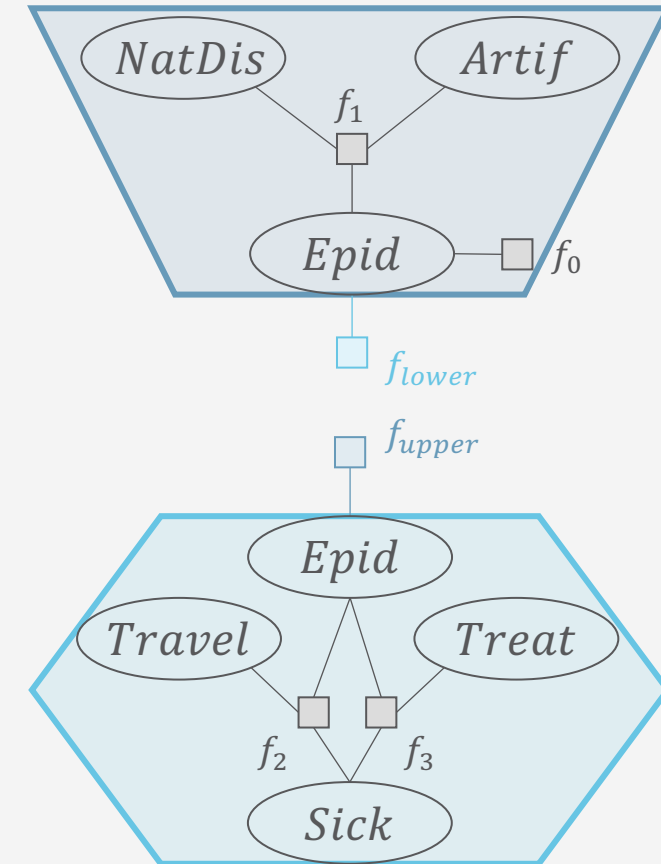
Independences to the Rescue?

- Global independences in the model:
 - $Treat, Travel, Sick \perp NatDis, Artif \mid Epid$
 - $Treat \perp Travel \mid \{Epid, Sick\}$
- If *Epid* is given, then the upper part is independent of the lower part and vice versa
- If additionally *Sick* is given, then the lower left part is independent of the lower right part and vice versa
- Queries could be answered independently from the other parts
- For that to be possible...
 - [Variant 1] the separators *Epid* or *Epid, Sick* would need to always be set with evidence
 - Unlikely; also would restrict the queries that could be asked
→ only partially helpful ↘



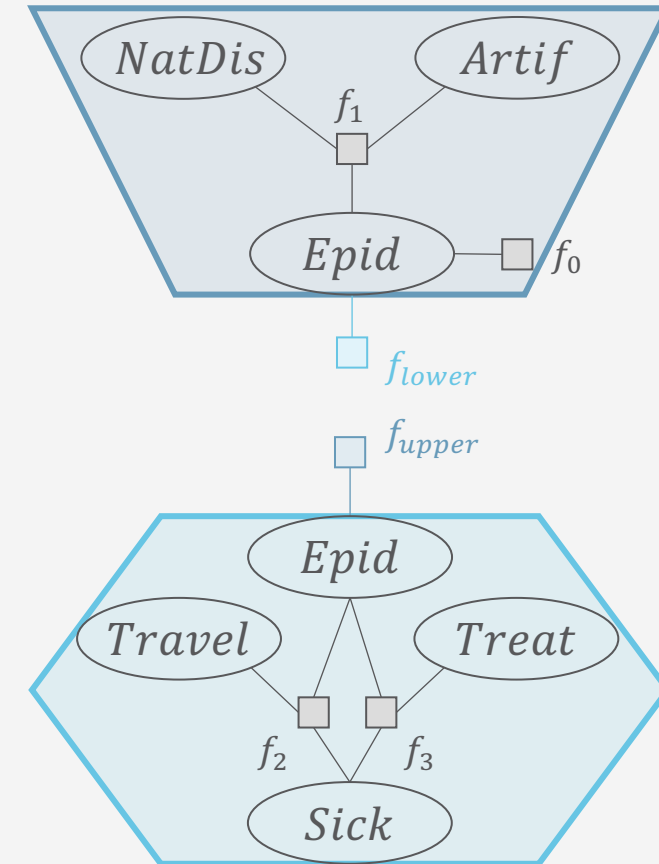
Independences to the Rescue?

- Global independences in the model:
 - $Treat, Travel, Sick \perp NatDis, Artif \mid Epid$
 - $Treat \perp Travel \mid \{Epid, Sick\}$
- If *Epid* is given, then the upper part is independent of the lower part and vice versa
- If additionally *Sick* is given, then the lower left part is independent of the lower right part and vice versa
- Queries could be answered independently from the other parts
- For that to be possible...
 - [Variant 2] the information from the other parts of the model beyond the separator would need to be collected beforehand
→ In all cases helpful ✓



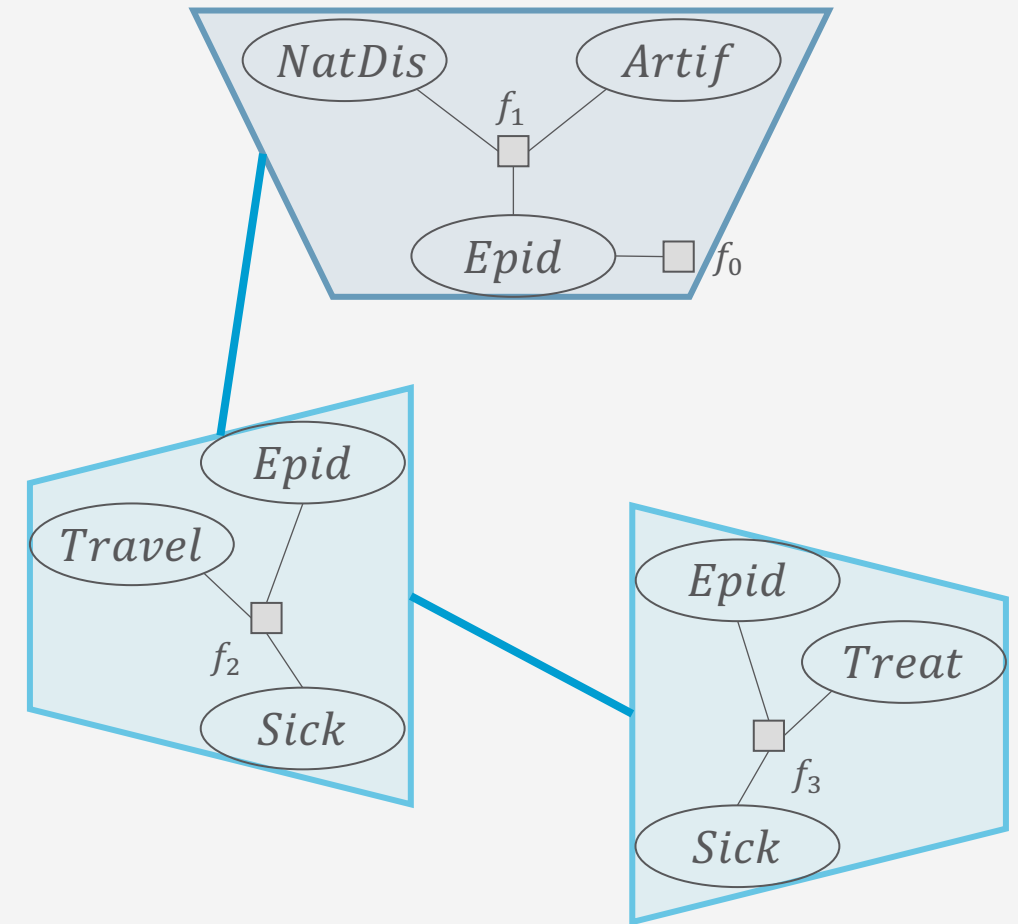
Independences to the Rescue!

- Is it this simple? – In a way, yes...
 - Using separators ensures that the information from the rest of the model does not reach the part twice and can be exchanged through messages over the separators
- Example:
 - Separator *Epid* between upper and lower part:
 - Message f_{lower} contains information about lower part: VE result of summing out *Treat*, *Travel*, *Sick* in f_2, f_3
 - Message f_{upper} contains information about upper part: VE result of summing out *Artif*, *NatDis* in f_1, f_0



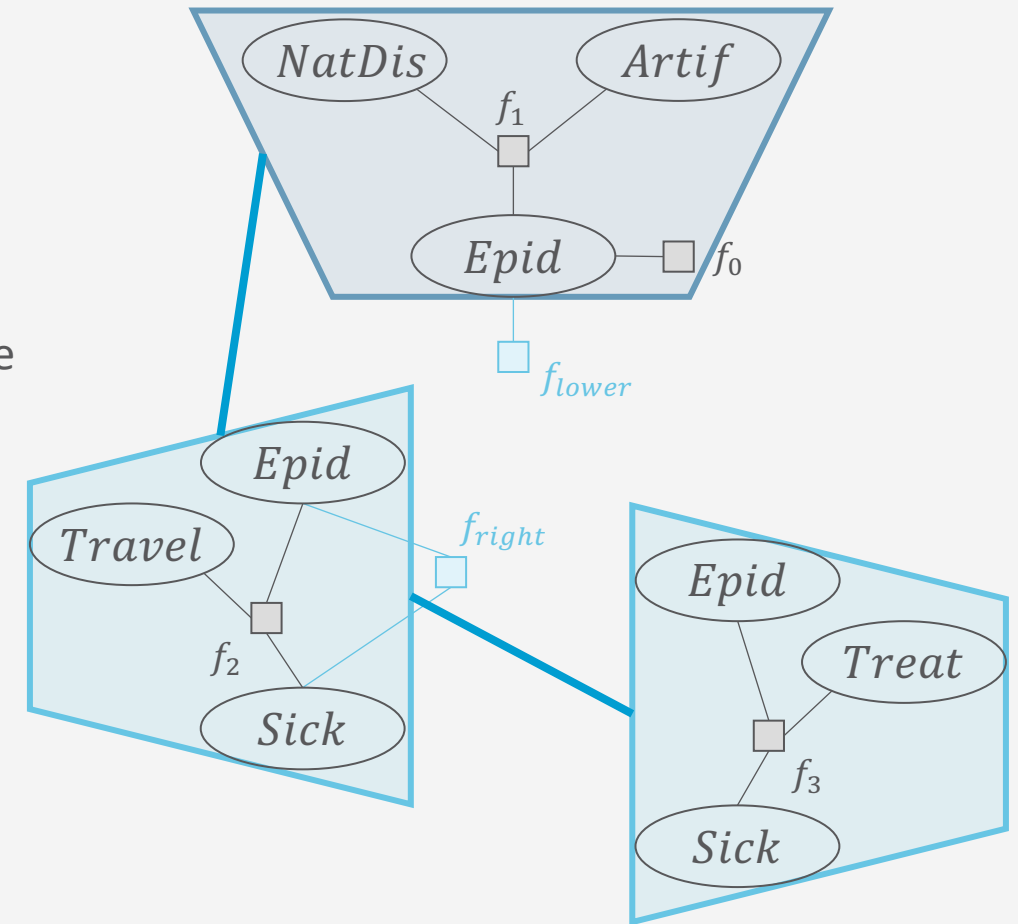
A Network of Clusters

- However, ...
 - Searching for the best interface for each query not efficient
 - Plus, we wanted to reduce duplicate computations
 - In example: Summing out *Travel* saved for upper part, but also needed for query on *Treat*
- Remedy
 - Network of *clusters* of random variables, as small as possible, with global independences among them
 - Example: Split lower cluster into two clusters



Messages in the Network

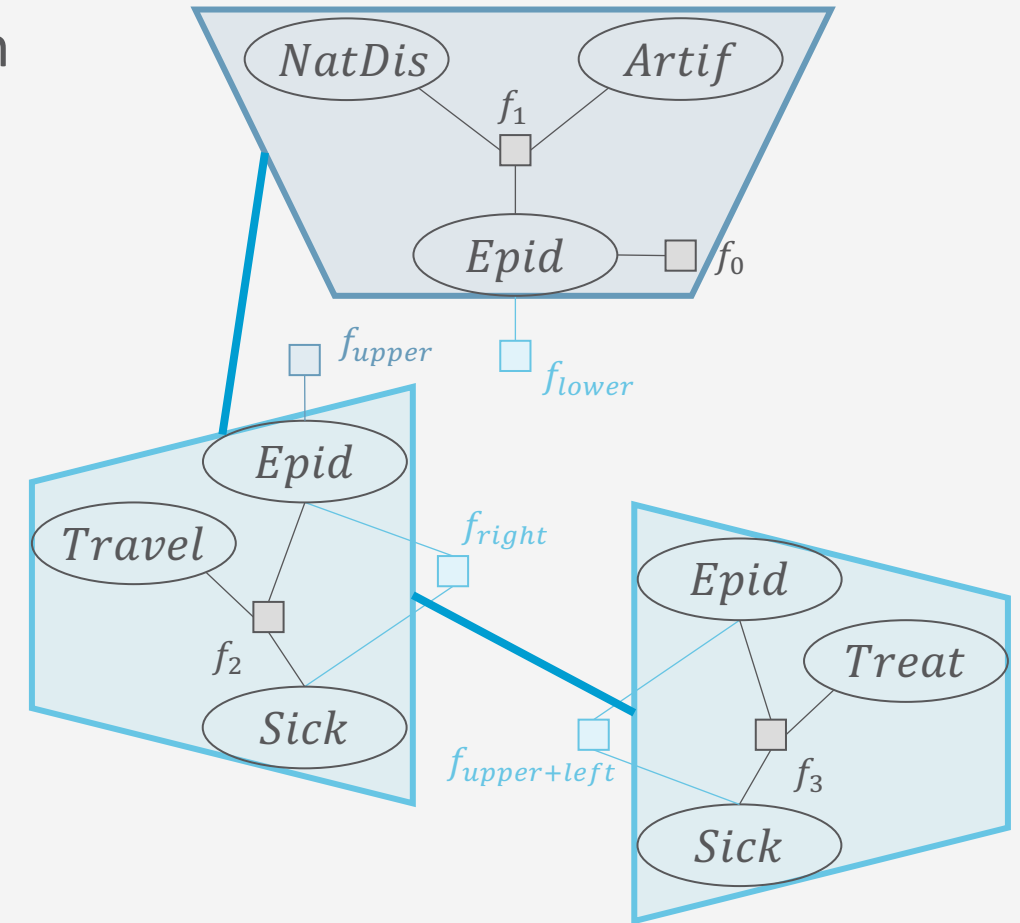
- To answer queries on the clusters, the clusters need to be independent
 - That is messages need to be available at each interface
 - How?
 - C_{upper} could request a message from C_{left}
 - For C_{left} to be able to calculate the message, it needs message from C_{right}
 - C_{left} requests message from C_{right}
 - C_{right} calculates and sends message, f_{right} , to C_{left}
 - Summing out $Treat$ in f_3
 - C_{left} calculates and send message, f_{lower} , to C_{upper}
 - Summing out $Travel, Sick$ in f_2, f_{right}
 - C_{upper} then has all information from the model to answer queries on its own cluster variables



Messages in the Network

Idea of the Junction Tree Algorithmus

- For every cluster to be able to answer queries, each cluster has to request these messages
- Send messages in two phases:
 - From one end to the other and back, i.e., two messages for each edge in the network
 - “one end” messages from C_{right} over C_{left} to C_{upper}
 - “back” messages: f_{upper} from C_{upper} to C_{left} and $f_{upper+left}$ from C_{left} to C_{right}
 - For f_{upper} : Sum out $NatDis, Artif$ in f_0, f_1
 - For $f_{upper+left}$: Sum out $Travel$ in f_{upper}, f_2
- For maximum parallelisability: Send messages from periphery to centre and back



VE for Message Passing in JT: Without Evidence and Normalisation

VE-JT(F, S, h)

while $\text{rv}(F) \setminus S \neq \emptyset$ **do**

$U \leftarrow \arg \min_R h(F)$

while $\exists f_1, f_2 \in F : U \in \text{rv}(f_1) \wedge \text{rv}(f_2)$ **do**

$F \leftarrow F \setminus \{f_1, f_2\} \cup \{\text{MULTIPLY}(f_1, f_2)\}$

$F \leftarrow F \setminus \{f\} \cup \{\text{SUM-OUT}(f, U)\}$

return F

Eliminate non-separator variables

▸ Choose next U to eliminate

▸ Multiply f_1, f_2 in F

▸ Sum out U in F