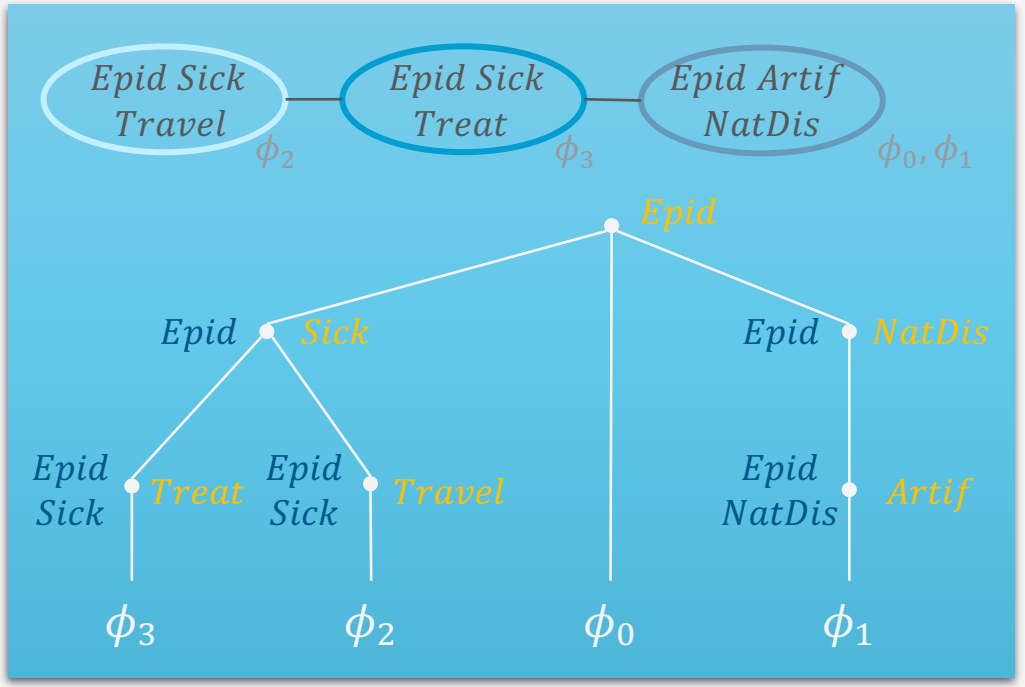
WWU
MÜNSTER

Exakte Inferenz in Episodischen PGMs

(a) Einzelanfragen: Variableneliminierung

Einführung in die
Künstliche Intelligenz



wissen.leben

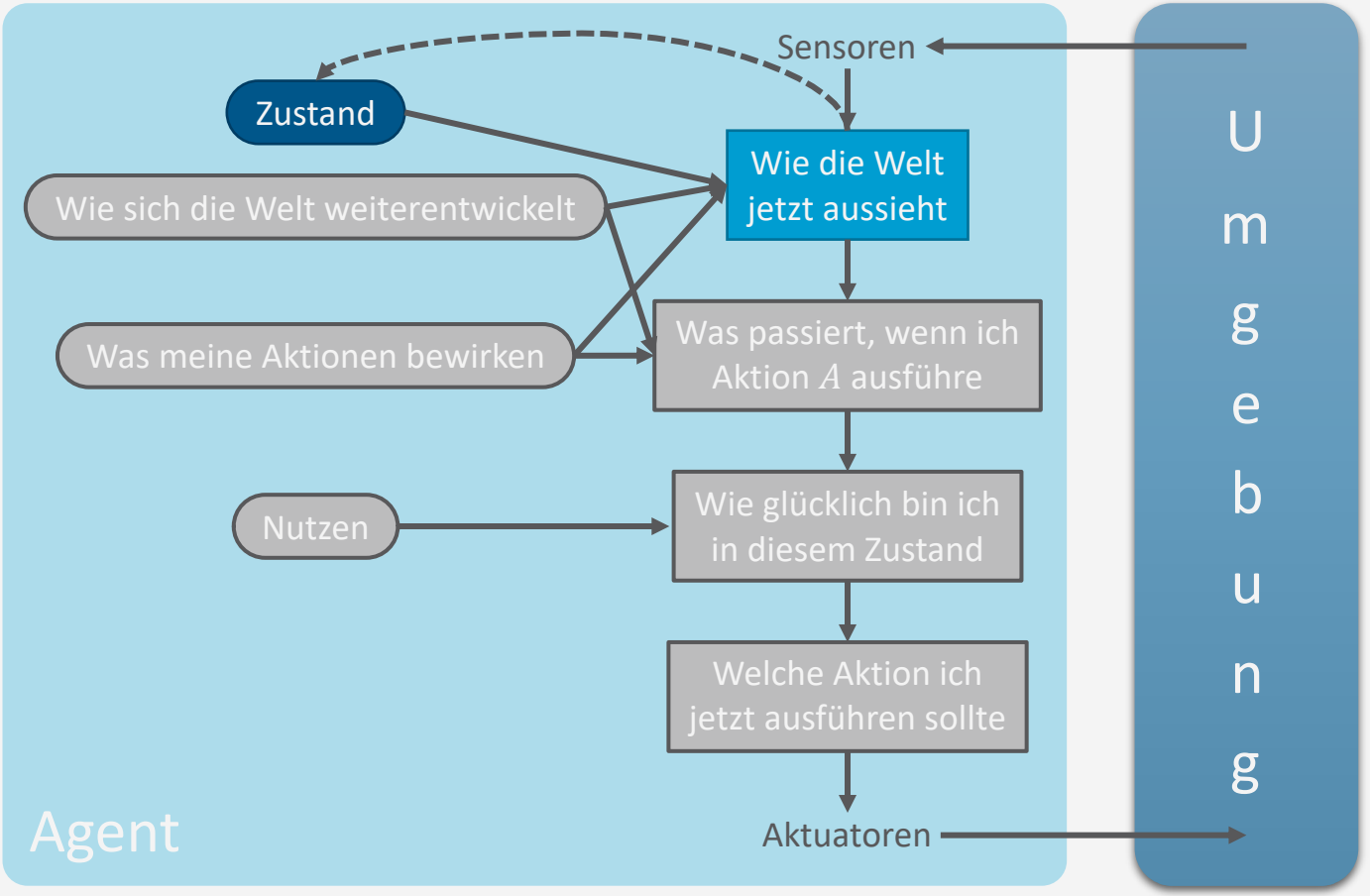
Tanya Braun
Arbeitsgruppe Data Science, Institut für Informatik

Inhalte

1. **Künstliche Intelligenz & Agenten**
   - Agentenabstraktion, Rationalität
   - Aufgabenumgebung
2. **Episodische PGMs**
   - Gerichtetes Modell: Bayes Netze (BNs)
   - Ungerichtete Modelle
3. **Exakte Inferenz in episodischen PGMs**
   - Wahrscheinlichkeits- und Zustandsanfragen
   - Direkt auf den Modellen, mittels Hilfsstrukturen
4. **Approximative Inferenz in episodischen PGMs**
   - Wahrscheinlichkeitsanfragen
   - Deterministische, stochastische Algorithmen
5. **Lernalgorithmen für episodische PGMs**
   - Bei (nicht) vollständigen Daten, (un)bekannter Struktur
6. **Sequentielle PGMs und Inferenz**
   - Dynamische BNs, Hidden-Markov-Modelle
   - filtering / prediction / hindsight Anfragen, wahrscheinlichste Zustandssequenz
   - Exakter, approximativer Algorithmus
7. **Entscheidungstheoretische PGMs**
   - Präferenzen, Nutzenprinzip
   - PGMs mit Entscheidungs- und Nutzenknoten
   - Berechnung der besten Aktion (Aktionssequenz)
8. **Abschlussbetrachtungen**

Einordnung der Vorlesung: *Modell- und nutzenbasierter Agent*

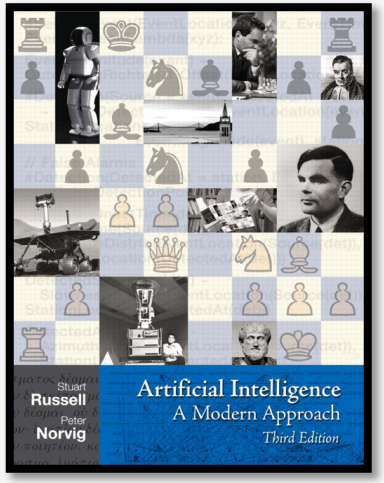
- Nachfolgende Themen der Vorlesung
  2. Episodische PGMs
  3. Exakte Inferenz in episodischen PGMs
  4. Approximative Inferenz in episodischen PGMs
  5. Lernalgorithmen für episodische PGMs
  6. Sequentielle PGMs und Inferenz
  7. Entscheidungstheoretische PGMs

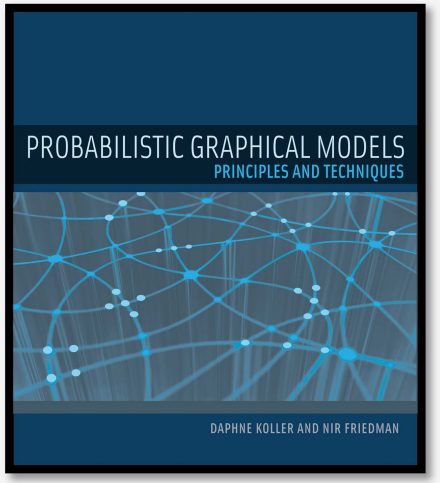


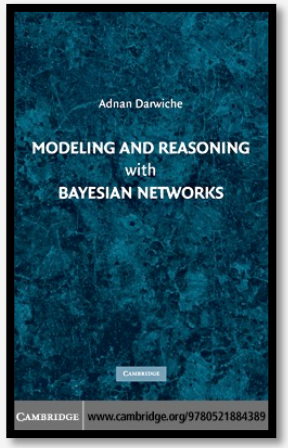
Literaturhinweise

Inhalte dieses Themenblocks werden in den folgenden Kapiteln der Vorlesungsbücher behandelt

- AIMA(de)
  - Kap. 14.4: Exakte Inferenz in Bayes Netzen
- PGM
  - Kap. 9: Variableneliminierung
    - Besonders: 9.3
  - Kap. 10: Exakte Inferenz: Cliquen-Bäume
  - Kap. 13: MAP Inferenz







- Wer gerne ein anderes Buch ausprobieren möchte (Fokus auf BNs):
  - Adnan Darwiche, *Modelling and Reasoning with Bayesian Networks*, 2009.

Überblick: 3. Exakte Inferenz in episodischen PGMs

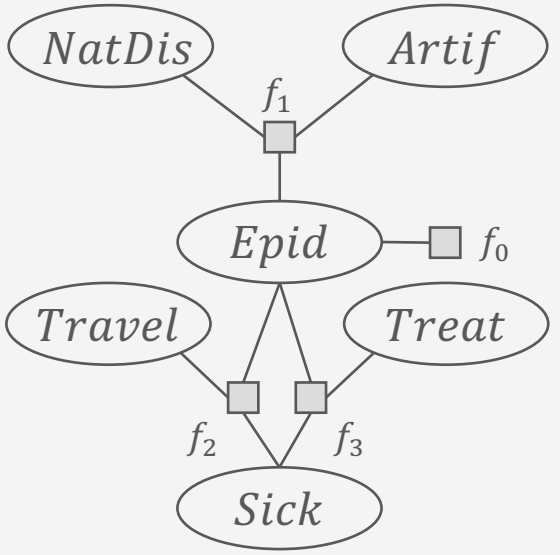
A. ***Einzelanfragen: Variableneliminierung (VE)***
   - Algorithmus, Operatoren für Wahrscheinlichkeitsanfragen
   - Dekompositionsbäume, Komplexität

B. *Multi-Anfragen: Junction Tree (Cliquen-Bäume) Algorithmus (JT)*
   - Cliquen, Junction Tree als Hilfsstruktur, Vorverarbeitung und Anfragebeantwortung
   - Zusammenhang mit VE, Komplexität
   - *Geschichtsstunde: Pearl's Probability Propagation (PP)* auf Polytree BNs

C. *Inferenzproblem Zustandsanfragen*
   - Ausprägungen: Most probable explanation (MPE) / maximum a posteriori Anfragen (MAP)
   - Semantik: Ausmaximieren statt aussummieren; max-out Operator in VE
   - Auswirkungen auf die Komplexität

Probabilistische Inferenz: Das Problem der Anfragenbeantwortung

- Erinnerung: Beantwortung von Anfragen in vollständigen gemeinsamen Verteilungen, indem Evidenz absorbiert und Nichtanfrage-Zufallsvariablen aussummiert werden
- Problem: Exponentiell abhängig von der Anzahl der Zufallsvariablen
- Beispiel:
  - 6 Boolesche Zufallsvariablen $\rightarrow 2^6 = 64$ Einträge in $P_{\boldsymbol{R}}$
  - Leere Anfrage $P(.) \rightarrow 6$ Variablen eliminieren
    - Nach und nach alle 64 Einträge durchgehen und aufaddieren
- Idee: Wenn eine Faktorisierung bekannt ist, dann diese für eine effizientere Aussummierung nutzen
- Fokus der weiteren Vorlesung:
  - Anfragen der Form $P(\boldsymbol{S} \mid \boldsymbol{t})$
    - Evidenz $\boldsymbol{t}$ = Menge von Beobachtungen; kann leer sein: $\boldsymbol{t} = \emptyset \rightarrow P(\boldsymbol{S})$
  - Algorithmen spezifiziert für Faktormodelle, gelten aber ebenso für BNs



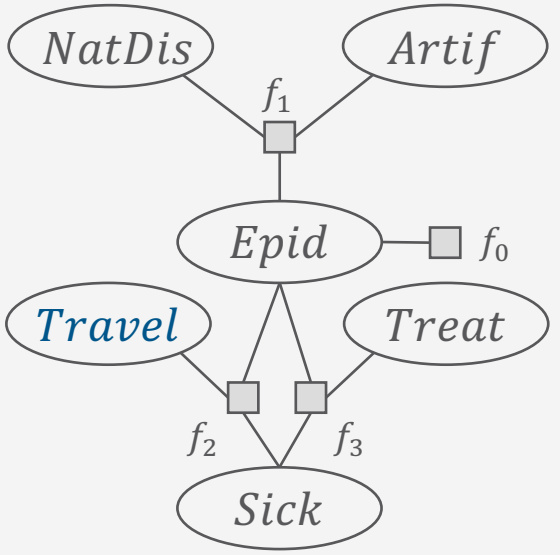
Variableneliminierung (VE)

- Grundidee VE: Evidenz absorbieren und Nichtanfrage-Variablen $\{R_1, \dots, R_m\} = \boldsymbol{R} \setminus \mathrm{rv}(\boldsymbol{S}, \boldsymbol{t})$ aussummieren unter Ausnutzung der Faktorisierung im Faktormodell F

$$P(\boldsymbol{S} \mid \boldsymbol{t}) = \frac{1}{P(\boldsymbol{t})} \sum_{v_1 \in \mathrm{Val}(R_1)} \dots \sum_{v_n \in \mathrm{Val}(R_m)} P_{\boldsymbol{R}}(R_1 = v_1, \dots, R_m = v_m, \boldsymbol{S}, \boldsymbol{t})$$

$$= \frac{1}{P(\boldsymbol{t})} \sum_{v_1 \in \mathrm{Val}(R_1)} \dots \sum_{v_n \in \mathrm{Val}(R_m)} \prod_{f \in F} f$$

  - Faktor aus Summen ausklammern, deren Argumente nicht von der Summe berührt werden
  - Teilen durch $P(\boldsymbol{t})$ = Normalisierung des Summenausdrucks $P(\boldsymbol{S}, \boldsymbol{t})$
- Beispiel: $P(Travel)$ in $F = \{f_i\}_{i=0}^{3}$
  - $\{Epid, NatDis, Artif, Sick, Treat\}$ aussummieren



Variableneliminierung (VE): Beispiel

So nicht ganz korrekt, da wir nur Werte zuweisen, wenn ein Argument keine Anfrage-Variable ist; genauer wäre:
$\pi_{\mathrm{rv}(f_i)}(E = e, N = n, A = a, S = s, Travel, T = t)$
- Projektion (Auswahl; π) der Eingaben von $P_{\boldsymbol{R}}$ auf die Argumente pro Faktor ($\mathrm{rv}(f_i)$)

$$P(Travel)$$

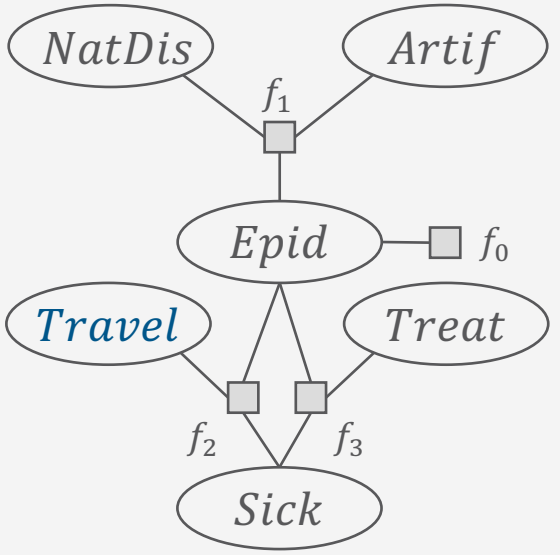
$$\propto \sum_{e \in \mathrm{Val}(E)} \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \sum_{s \in \mathrm{Val}(S)} \sum_{t \in \mathrm{Val}(T)} P_{\boldsymbol{R}}(E = e, N = n, A = a, S = s, Travel, T = t)$$

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

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

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

Summen sogar unabhängig voneinander
- Könnten parallel ausgerechnet werden



$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

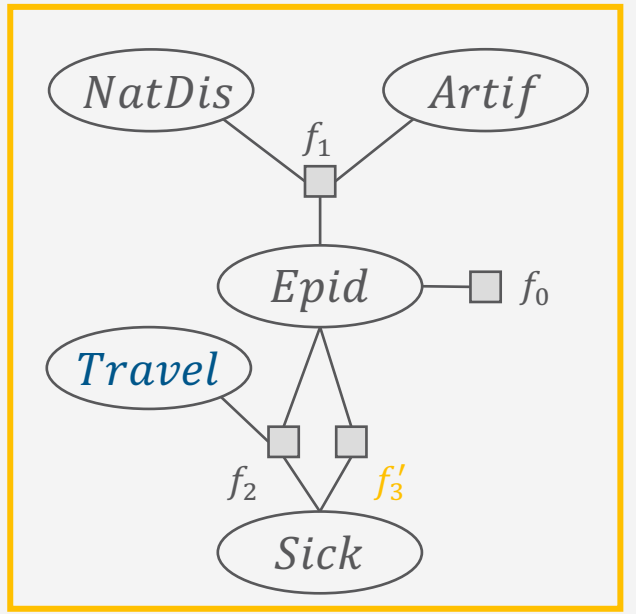
$$P(Travel)$$

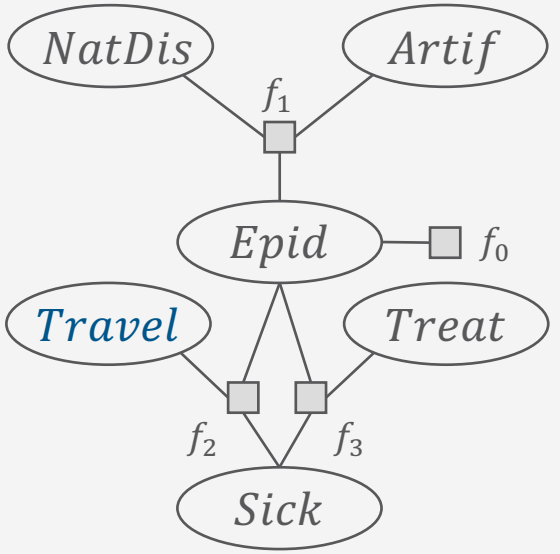
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a) \sum_{s \in \mathrm{Val}(S)} \phi_2(Travel,e,s) \sum_{t \in \mathrm{Val}(T)} \phi_3(e,s,t)$$

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

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	*Sick*	ϕ_3'
false	*false*	6
false	*true*	5
true	*false*	9
true	*true*	8





$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

$P(Travel)$

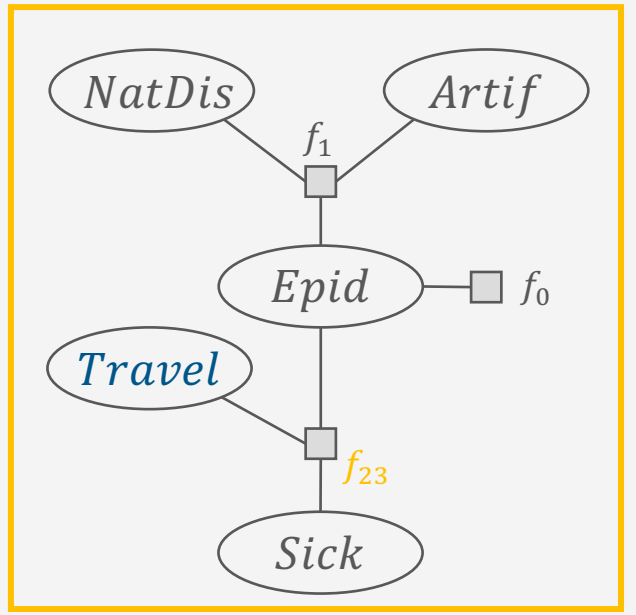
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a) \sum_{s \in \mathrm{Val}(S)} \phi_2(Travel,e,s)\phi_3'(e,s)$$

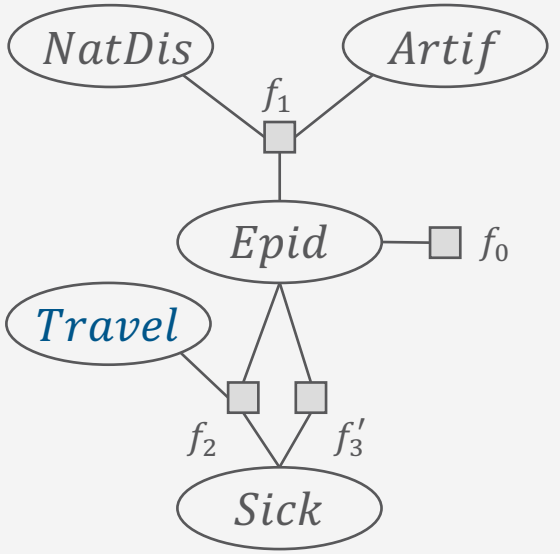
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a) \sum_{s \in \mathrm{Val}(S)} \phi_{23}(Travel,e,s)$$

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

Epid	Sick	ϕ_3'
false	false	6
false	true	5
true	false	9
true	true	8

Travel	Epid	Sick	ϕ_{23}
false	false	false	$20 \cdot 6 = 120$
false	false	true	$24 \cdot 5 = 120$
false	true	false	$5 \cdot 9 = 45$
false	true	true	$6 \cdot 8 = 48$
true	false	false	$28 \cdot 6 = 168$
true	false	true	$8 \cdot 5 = 40$
true	true	false	$7 \cdot 9 = 63$
true	true	true	$2 \cdot 8 = 16$





$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

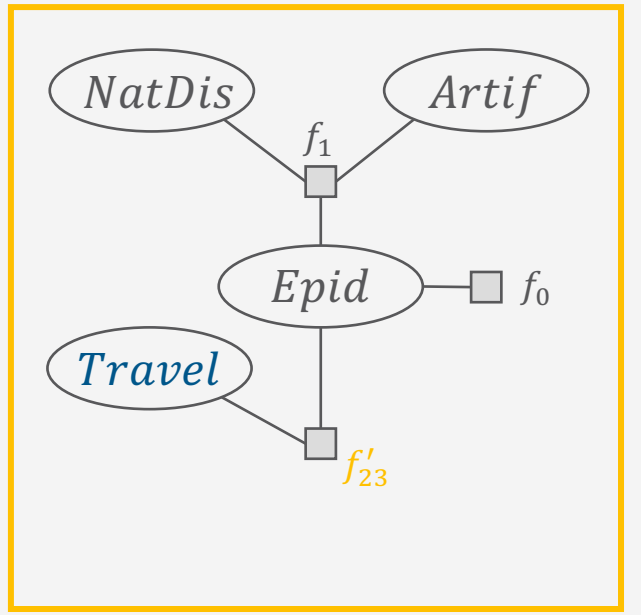
$$P(Travel)$$

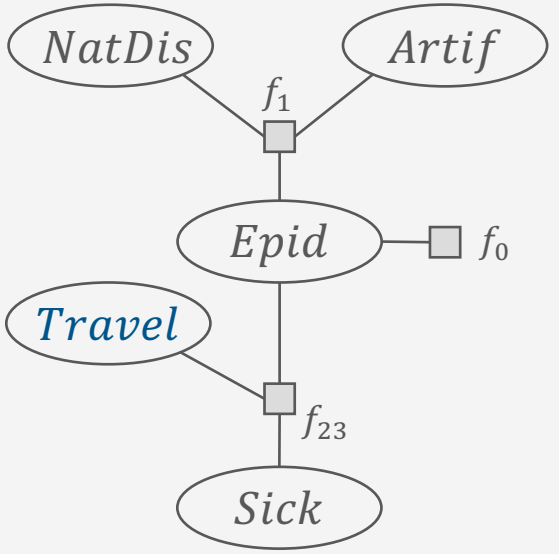
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a) \sum_{s \in \mathrm{Val}(S)} \phi_{23}(Travel,e,s)$$

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

Travel	Epid	Sick	ϕ_{23}
false	false	false	120
false	false	true	120
false	true	false	45
false	true	true	48
true	false	false	168
true	false	true	40
true	true	false	63
true	true	true	16

Travel	Epid	ϕ'_{23}
false	false	240
false	true	93
true	false	208
true	true	79



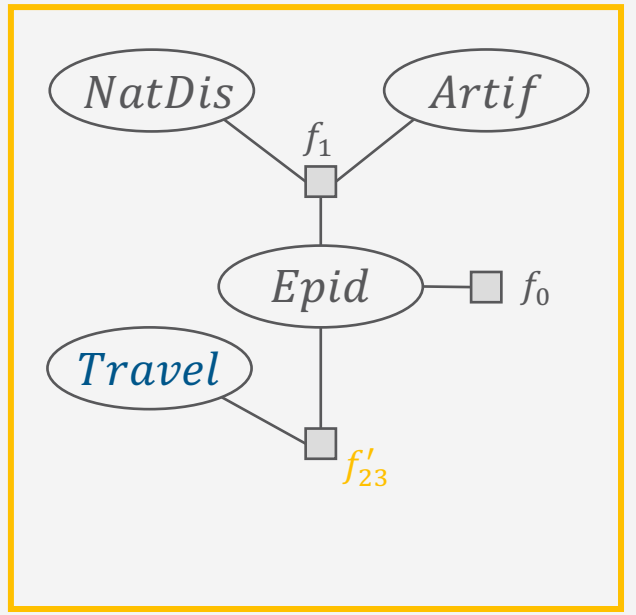


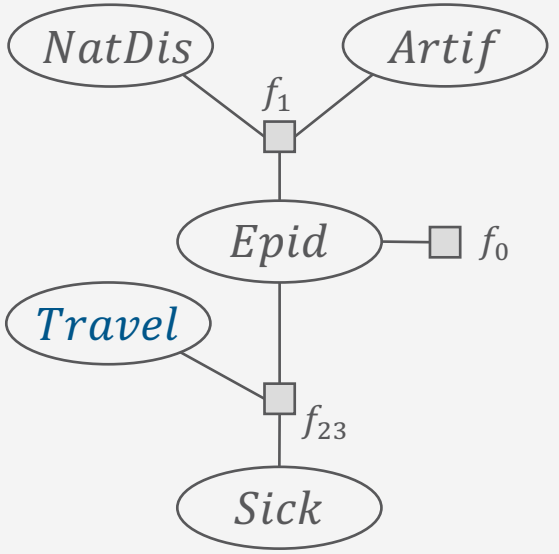
$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

$$P(Travel)$$

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

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





$$P(Travel)$$

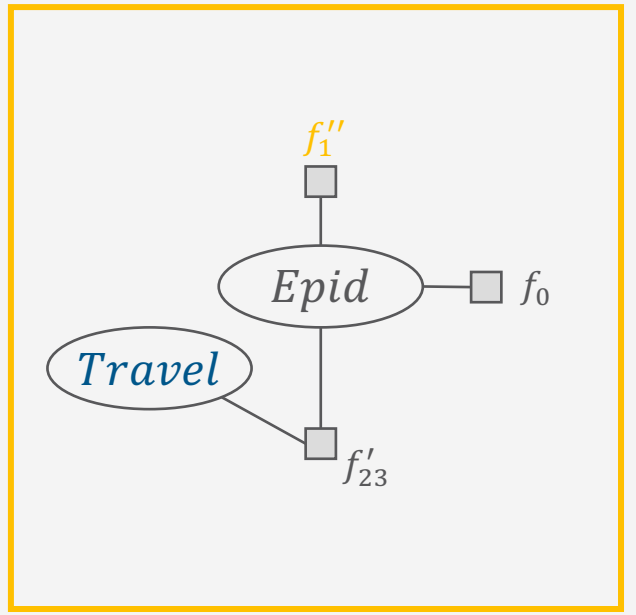
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi'_{23}(Travel, e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e, n, a)$$

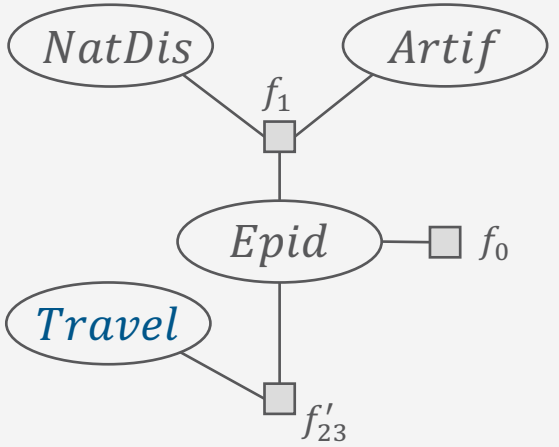
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi'_{23}(Travel, e)\phi''_1(e)$$

Epid	NatDis	Artif	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

Epid	NatDis	ϕ'_1
false	false	14
false	true	4
true	false	11
true	true	6

Epid	ϕ''_1
false	18
true	17





$P(Travel)$

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

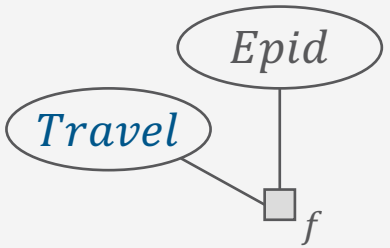
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi(Travel, e)$$

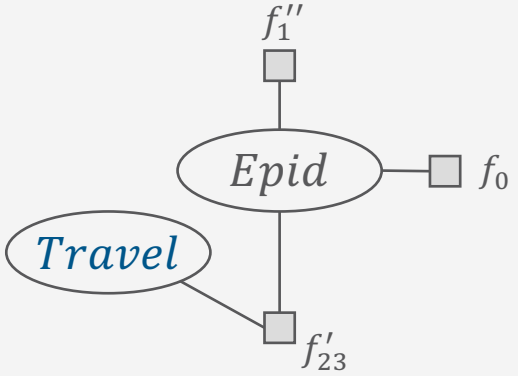
$Travel$	$Epid$	ϕ'_{23}
$false$	$false$	240
$false$	$true$	93
$true$	$false$	208
$true$	$true$	79

$Epid$	ϕ''_1
$false$	18
$true$	17

$Epid$	ϕ_0
$false$	50
$true$	1

$Travel$	$Epid$	ϕ
$false$	$false$	$240 \cdot 18 \cdot 50 = 216.000$
$false$	$true$	$93 \cdot 17 \cdot 1 = 1.581$
$true$	$false$	$208 \cdot 18 \cdot 50 = 187.200$
$true$	$true$	$79 \cdot 17 \cdot 1 = 1.343$





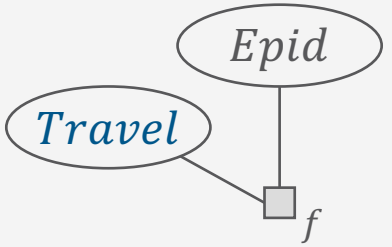
$$P(Travel)$$

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

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

$Travel$	$Epid$	ϕ
$false$	$false$	216.000
$false$	$true$	1.581
$true$	$false$	187.200
$true$	$true$	1.343

$Travel$	ϕ'
$false$	217.581
$true$	188.543



$P(Travel)$

$\propto \phi'(Travel)$

$= \phi^n(Travel)$

$= P(Travel)$

$Travel$	ϕ
$false$	217.581
$true$	188.543

$Travel$	ϕ^n
$false$	$\frac{217.581}{217.581 + 188.543} = \frac{217.581}{406.124} = 0.54$
$true$	$\frac{188.543}{217.581 + 188.543} = \frac{188.543}{406.124} = 0.46$

$$P(Travel)$$

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

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

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

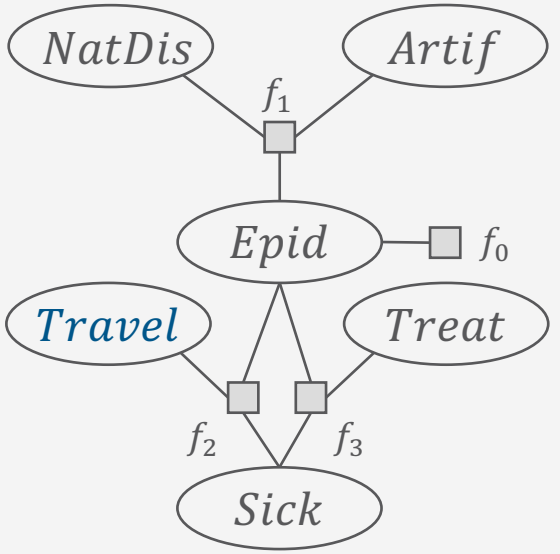
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_{23}'(Travel,e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a)$$

$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_{23}'(Travel,e)\phi_1''(e) = \sum_{e \in \mathrm{Val}(E)} \phi(Travel,e) = \phi'(Travel)$$

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

$$= P(Travel)$$

Unsere Zwischenergebnisse wurden nie größer als $2^3 < 2^6$



$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

VE mit Evidenz

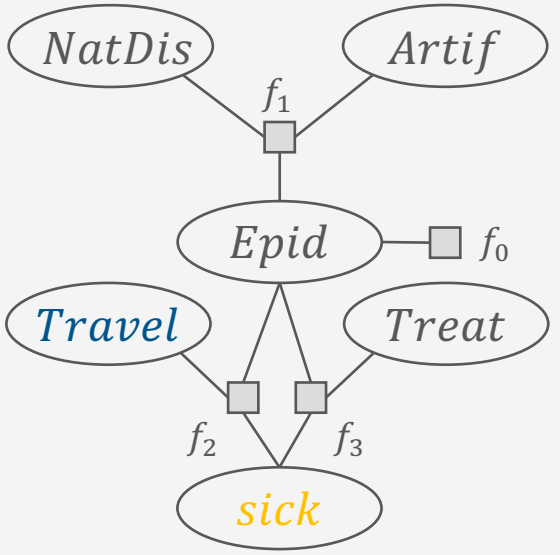
- Im Vergleich zu Marginalanfragen kommt Absorbierung als erster Schritt bei Anfragen mit Evidenz ($\boldsymbol{t} \neq \emptyset$) hinzu
  - Der Rest bleibt gleich: Faktoren aus Summen ausklammern, deren Argumente nicht von der Summe berührt werden
- Vorgehen zur Absorbierung von einer Beobachtung $T = t$:

a. Bei $T \neq t$ Wahrscheinlichkeiten auf 0 setzen
b. Zeilen mit Wahrscheinlichkeit 0 fallen lassen
c. Evidenzvariable fallen lassen

Kürzer gefasst:

i. Zeilen mit $T \neq t$ fallen lassen
ii. Evidenzvariable fallen lassen

- Was heißt das im Rahmen von VE?
  - Jeder Faktor f mit $T \in \mathrm{rv}(f)$ muss die Evidenz $T = t$ absorbieren
  - Beispiel: $P(Travel \mid sick) \rightarrow f_2, f_3$ müssen $sick$ absorbieren



VE mit Evidenz: Beispiel

- Absorbieren von $sick$ in f_2, f_3

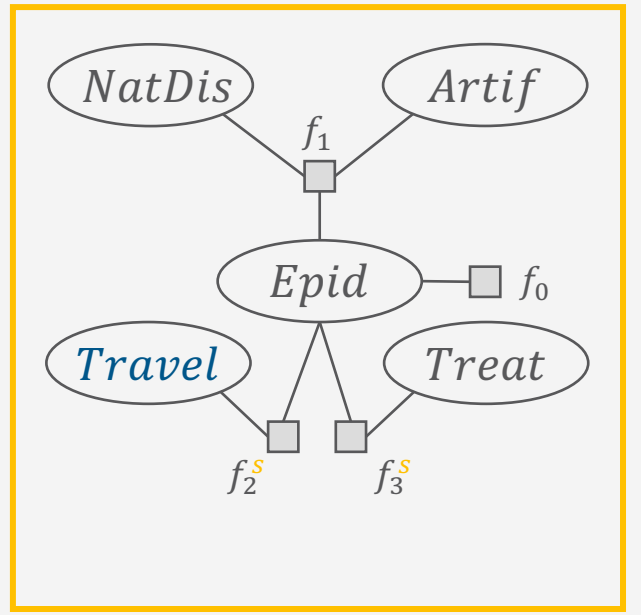
Quasi aus Modell $F = \{f_0, f_1, f_2, f_3\}$ mit Anfrage $P(Travel \mid sick)$
das Modell $F' = \{f_0, f_1, f_2^s, f_3^s\}$ mit Anfrage $P(Travel)$ gemacht

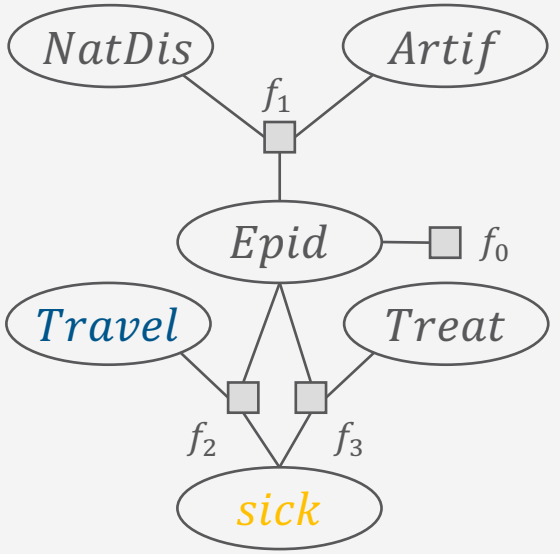
Epid	Sick	Treat	ϕ_3
false	false	false	5
false	false	true	1
false	true	false	3
false	true	true	2
true	false	false	5
true	false	true	4
true	true	false	1
true	true	true	7

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





VE mit Evidenz: Beispiel

$$P(Travel \mid sick)$$

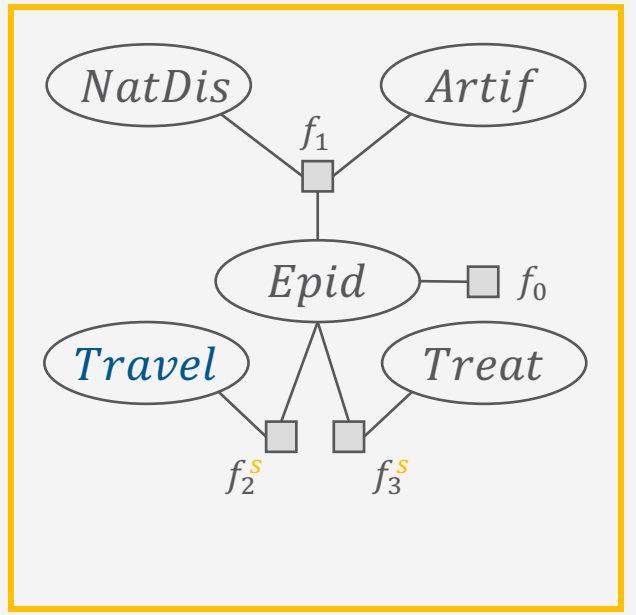
$$\propto \sum_{e \in \mathrm{Val}(E)} \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \sum_{t \in \mathrm{Val}(T)} P_{\boldsymbol{R}}(E = e, N = n, A = a, sick, Travel, T = t)$$

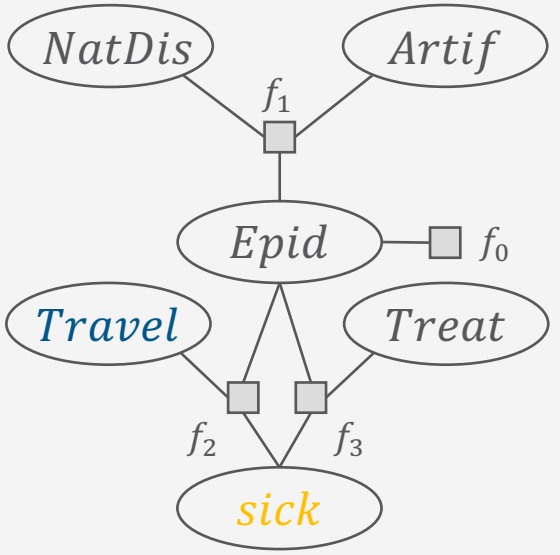
$$\propto \sum_{e \in \mathrm{Val}(E)} \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \sum_{t \in \mathrm{Val}(T)} \prod_{i=0}^{3} \phi_i(\boldsymbol{R}_i = \boldsymbol{r}_i)$$

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

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

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





$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, T \triangleq Treat$

VE mit Evidenz: Beispiel

$$P(Travel \mid sick)$$

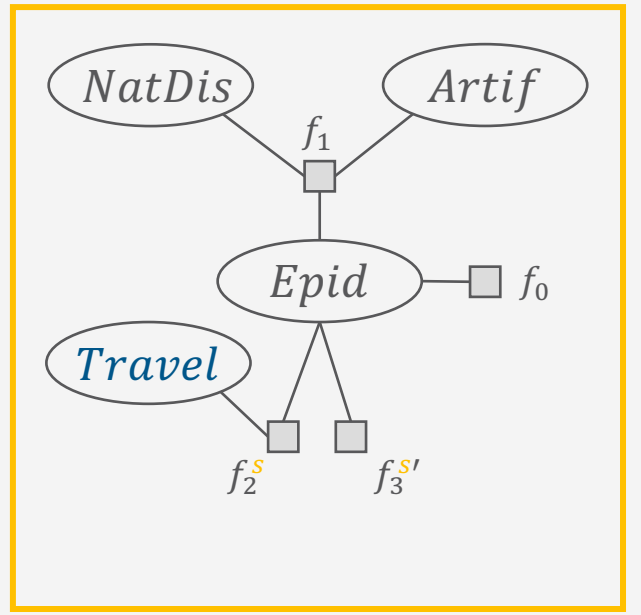
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_2^s(Travel, e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e, n, a) \sum_{t \in \mathrm{Val}(T)} \phi_3^s(e, t)$$

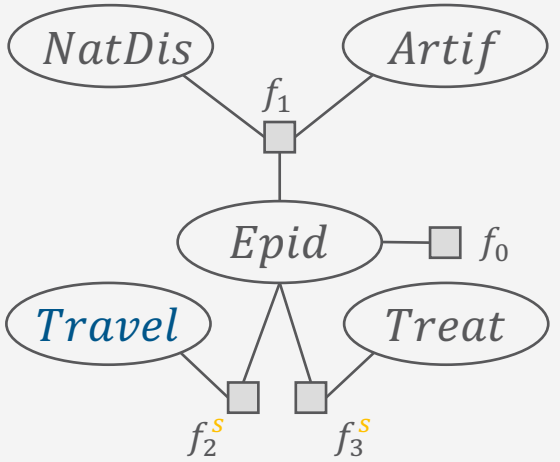
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_2^s(Travel, e)\phi_3^{s'}(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e, n, a)$$

$Epid$	$Treat$	ϕ_3^s
$false$	$false$	3
$false$	$true$	2
$true$	$false$	1
$true$	$true$	7

+

$Epid$	$\phi_3^{s'}$
$false$	5
$true$	8





VE mit Evidenz: Beispiel

$P(Travel \mid sick)$

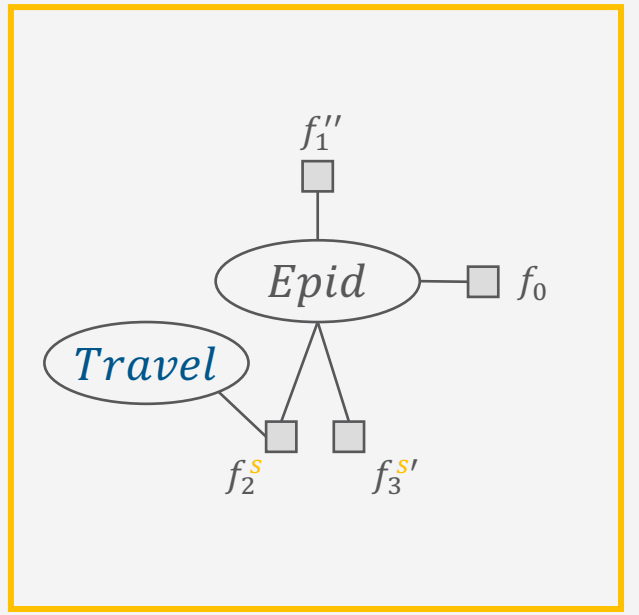
Wie vorher

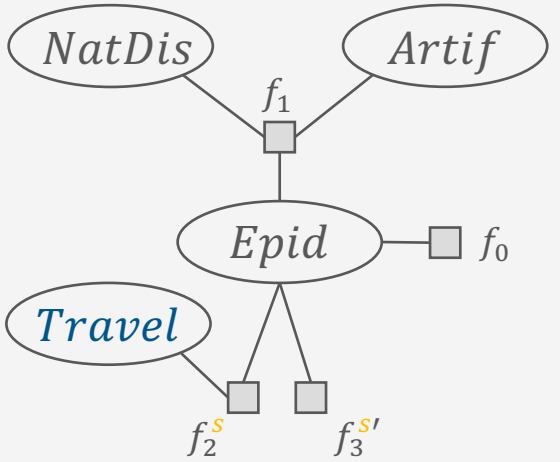
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_2^s(Travel, e)\phi_3^{s\prime}(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e, n, a)$$

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

$Epid$	$NatDis$	$Artif$	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

$Epid$	ϕ_1''
$false$	18
$true$	17



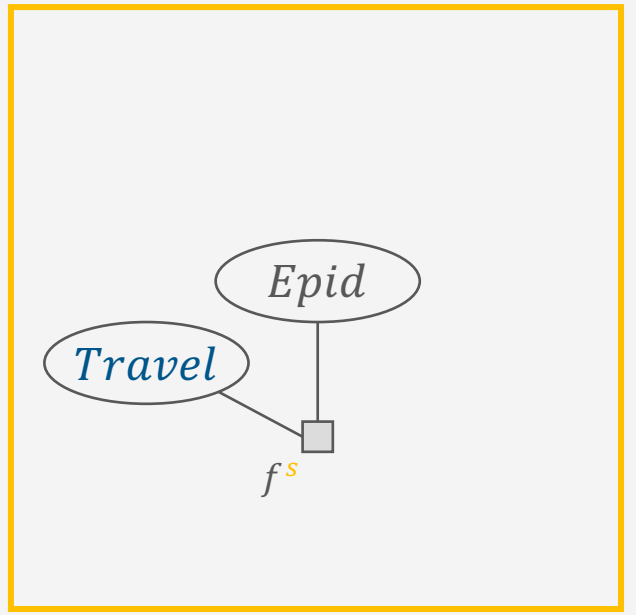


VE mit Evidenz: Beispiel

$P(Travel \mid sick)$

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

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



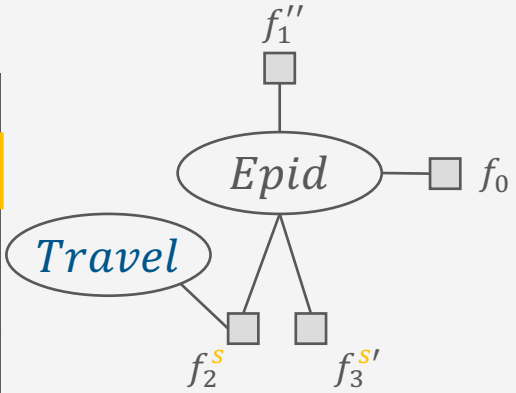
$Travel$	$Epid$	ϕ_2^s
$false$	$false$	24
$false$	$true$	6
$true$	$false$	8
$true$	$true$	2

$Epid$	ϕ_0
$false$	50
$true$	1

$Epid$	$\phi_3^{s\prime}$
$false$	5
$true$	8

$Epid$	ϕ_1''
$false$	18
$true$	17

$Travel$	$Epid$	ϕ
$false$	$false$	$24 \cdot 50 \cdot 5 \cdot 18 = 108.000$
$false$	$true$	$6 \cdot 1 \cdot 8 \cdot 17 = 816$
$true$	$false$	$8 \cdot 50 \cdot 5 \cdot 18 = 36.000$
$true$	$true$	$2 \cdot 1 \cdot 8 \cdot 17 = 272$



VE mit Evidenz: Beispiel

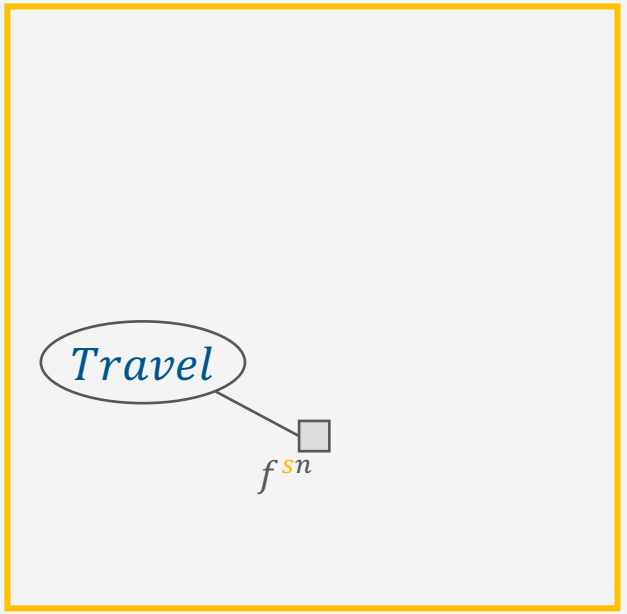
$$P(Travel \mid sick)$$

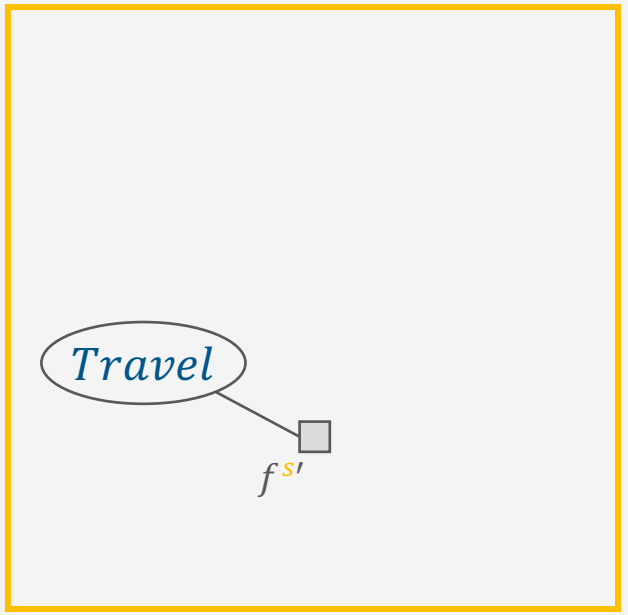
$$\propto \sum_{e \in \mathrm{Val}(E)} \phi^{s}(Travel, e)$$

$$\propto \phi^{s'}(Travel)$$

$$= \phi^{sn}(Travel)$$

$$= P(Travel \mid sick)$$





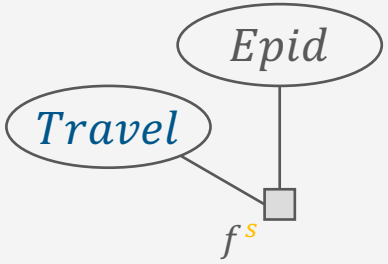
Travel	Epid	ϕ^{s}
false	false	108.000
false	true	816
true	false	36.000
true	true	272

+

+

Travel	$\phi^{s'}$
false	108.816
true	36.272

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

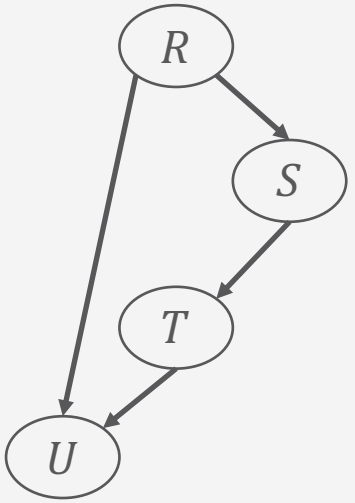


Variableneliminierung in BNs

- In BNs, VE startet mit Wahrscheinlichkeitsverteilungen
- Zwischenergebnisse möglicherweise als bedingte Wahrscheinlichkeitsverteilungen im BN interpretierbar, aber nicht immer
  - Beispiel: BN B über R, S, T, U mit $P(R), P(S \mid R), P(T \mid S), P(U \mid R, T)$
    - Eliminierung von X:

$$\phi(S, T, U) = \sum_{r \in \mathrm{Val}(R)} P(r) \cdot P(S \mid r) \cdot P(U \mid r, T)$$

    - $\phi(S, T, U)$ stellt keine bedingte Wahrscheinlichkeitsverteilung aus B dar

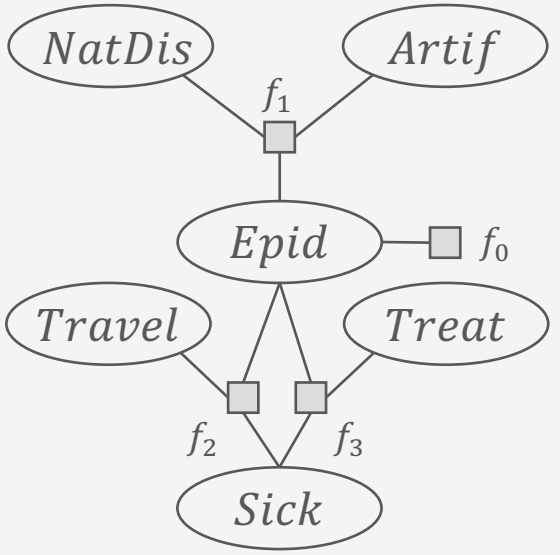


- In der Regel werden CPDs nach Rechenoperationen als Faktoren ϕ referenziert
  - Beispiel:

$$P(U) = \sum_r P(r) \sum_t P(U \mid r, t) \sum_s P(t \mid s) \cdot P(s \mid r) = \sum_r P(r) \sum_t P(U \mid r, t) \sum_s \phi(s, t, r) = \cdots$$

VE Operatoren

- Genutzte Operatoren in den Rechnungen
  - Aussummierung
    - Vorbedingung zum Aussummieren einer Variable: Variable darf nur in einem Faktor vorkommen
  - Multiplikation
    - Um Vorbedingung des Aussummierens zu erfüllen
  - Absorbierung
    - Um Beobachtungen zu verarbeiten
- Formale Definitionen folgen
  - Wozu?
    - Mittels Vor- und Nachbedingungen Korrektheit nachweisen
    - Grundlage für eine korrekte Implementierung



Aussummieren von Zufallsvariablen: Formale Definition

- Operator: SUM-OUT
  - Inputs:
    - Faktor $f = \phi(R_1, \dots, R_n) \in F$
    - Variable $R \in \{R_1, \dots, R_n\}$ an Position i zum Aussummieren
  - Vorbedingung: $\forall f' \in F \setminus \{f\}: R \notin \mathrm{rv}(f')$
  - Output: Faktor $\phi'(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$
    - Für alle möglichen Werte $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n$ von $R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n$
      - I.e., $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n \in \mathrm{Val}(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$

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

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

Multiplikation von Faktoren: Formale Definition

- Operator: MULTIPLY
  - Inputs:
    - Faktor $f_1 = \phi_1(R_1, \dots, R_n) \in F$
    - Faktor $f_2 = \phi_2(S_1, \dots, S_m) \in F$
  - Vorbedingung: *keine*
  - Output: Faktor $\phi(T_1, \dots, T_k)$
    - $\{(T_1, \dots, T_k)\} = \{(R_1, \dots, R_n)\} \bowtie \{(S_1, \dots, S_m)\}$ (geordnete Vereinigung)
    - Für alle möglichen Werte $t_1, \dots, t_k$ von $T_1, \dots, T_k$, i.e., $t_1, \dots, t_k \in \text{Val}(T_1, \dots, T_k)$, mit
      - $r_1, \dots, r_n = \pi_{R_1,\dots,R_n}(t_1, \dots, t_k)$ und $s_1, \dots, s_m = \pi_{S_1,\dots,S_m}(t_1, \dots, t_k)$ (Auswahl passend der Argumente)

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

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

Absorbieren von Beobachtungen: Formale Definition

- Operator: ABSORB
  - Inputs:
    - Faktor $f = \phi(R_1, \dots, R_n) \in F$
    - Zufallsvariable $R \in \{R_1, \dots, R_n\}$ an Position i
    - Faktor $f^r = \phi(R)$ mit Abbildungen $r \longmapsto 1$ und $r' \longmapsto 0\ \forall r' \neq r \in \mathrm{Val}(R)$ für Beobachtung $R = r$,
  - Vorbedingung: *keine*
  - Output: Faktor $\phi'(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$
    - Für alle möglichen Werte $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n$ von $R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n$
      - I.e., $r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_n \in \mathrm{Val}(R_1, \dots, R_{i-1}, R_{i+1}, \dots, R_n)$, mit

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

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

VE Algorithmus mit Eliminationsreihenfolge

- Eingabe:
  - Modell $F = \{f_i\}_{i=1}^{n}$
  - (Anfrageterme $\boldsymbol{S}$)
    - Werden im Algorithmus nicht gebraucht
  - Evidenz $\boldsymbol{t}$, gespeichert in Faktoren $\{\phi_t(T_t)\}_{t=1}^{|\boldsymbol{t}|}$
  - Reihenfolge $\mathcal{U} = (U_1, \dots, U_m)$ zur Aussummierung
    - $\{U_1, \dots, U_m\} = \mathrm{rv}(F) \setminus \mathrm{rv}(\boldsymbol{S}) \setminus \mathrm{rv}(\boldsymbol{t})$
- Ausgabe:
  - Faktor $f = \phi(\boldsymbol{S})$
    - Beinhaltet die (bedingte) Wahrscheinlichkeitsverteilung $P(\boldsymbol{S} \mid \boldsymbol{t})$

VE Algorithmus mit Eliminationsreihenfolge

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

for $t = 1, \dots, m$ **do** — Evidenz behandeln

  while $\exists f \in F : T_t \in \mathrm{rv}(f)$ **do**

    $F \leftarrow F \setminus \{f\} \cup \{\text{ABSORB}(f, T, \phi_t(T))\}$ ▹ Absorbiere $\phi_t(T_t)$ in F

for $i = 1, \dots, \mathrm{len}(\mathcal{U})$ **do** — Nichtanfrage-Variablen eliminieren

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

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

  $F \leftarrow F \setminus \{f\} \cup \{\text{SUM-OUT}(f, U_i)\}$ ▹ Summiere U_i aus F aus

while $\exists f_1, f_2 \in F$ **do** ▹ Multipliziere f_1, f_2 in F, bis $|F| = 1$

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

Normalisiere die Potentiale in $f \in F$ — Normalisieren

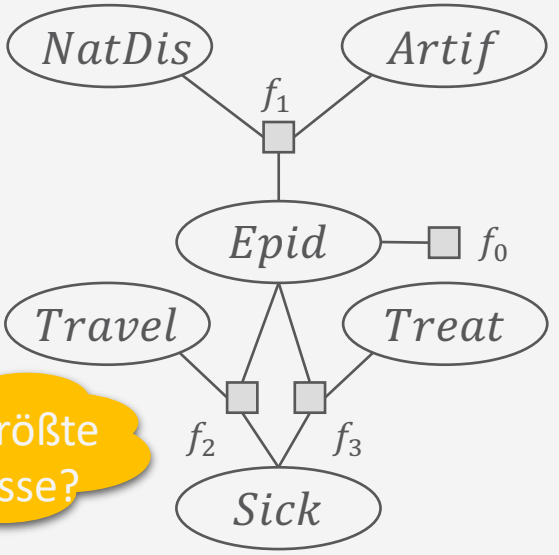
return f

VE

Reihenfolge der Operationen

- Reihenfolge $(U_1, \dots, U_m)$ zur Aussummierung
  → Woher nehmen, wenn nicht stehlen?

- Ziel: So aussummieren, dass die Zwischenergebnisse möglichst klein bleiben
  - Vorbedingung der Aussummierung kann Multiplikationen erfordern, die CPDs/Faktoren häufig größer machen und Zufallsvariablen verbindet, die vorher keine direkten Nachbarn waren
  - Beispiel:
    - Eliminationsreihenfolge im vorherigen Beispiel: $Treat, Sick, Artif, NatDis, Epid$
      - Größtes Zwischenergebnis: 2^3
    - Andere Reihenfolge: $Epid, Treat, Sick, Artif, NatDis$



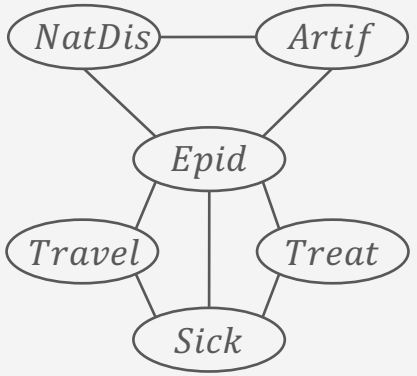
Wie groß ist das größte Zwischenergebnisse?

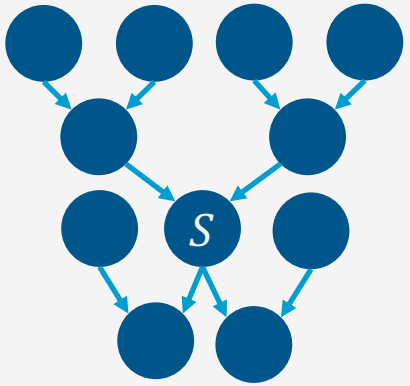
Reihenfolge der Operationen

- Optimale Reihenfolge in beliebigen Graphen finden nicht trivial
  - $\mathcal{NP}$-vollständig [ohne Beweis]
    - So schwer wie das Anfrageproblem selbst
  - Zum Beispiel über Tiefensuche finden
    - Suchbaum als Darstellung aller möglicher Eliminierungsfolgen in den Blättern mit Größe $O(n!), n = |\boldsymbol{R}|$
    - Trockendurchlauf von VE
      - Keine Rechnungen, sondern quasi reine Betrachtung der entstehenden Signaturen der Faktoren / Zwischenergebnisse um größte Anzahl an Argumenten identifizieren
    - Pruning möglich (ändert aber nichts an der Schwere des Problems), z.B.
      - Momentan beste Folge mit maximaler Größe o speichern
      - Suche in einem Pfad abbrechen, wenn die aktuelle Größe $x > o$

Reihenfolge der Operationen

- Spezialfälle
  - Chordale Graphen
    - Cliquen-weise eliminieren: Variablen, die nicht in anderen Cliquen vorkommen eliminieren, Zwischenergebnisse haben nicht mehr Variablen als Clique
  - Polytree BNs (ebenfalls chordal)
    - Eliminationsfolge bei *einer* Anfragevariable S, i.e., $|\boldsymbol{S}| = 1$, finden:
      - Polytree ungerichtet betrachten
      - S als Wurzel definieren
      - Kanten von der Wurzel weg ausrichten
      - Topologische Sortierung bauen
      - Zufallsvariablen in umgekehrter Reihenfolge zur topologischen Sortierung eliminieren
    - Idee: Von den Blättern bzw. Wurzeln aus starten, nach innen (hin zu S) rein eliminieren





Reihenfolge der Operationen

- Viel Forschungsarbeit in die Ausarbeitung von Heuristiken geflossen
- *Online Greedy* Variante:
  - Variable R als nächstes aussummieren, deren Zwischenergebnis am kleinsten ist
  - Quasi Ein-Schritt-VE-Simulation:
    - Für jedes zu eliminierende R
      - Menge der Faktoren F_R finden, die R enthalten, i.e., $R \in \mathrm{rv}(f), f \in F_R$ (*Herstellung der* SUM-OUT *Vorbedingung*)
      - Anzahl der Argumente nehmen $|\mathrm{rv}(F_R)|$ (Größe des Zwischenergebnisses vor dem Aussummieren)
    - Entscheidungskriterium: $\arg\min_R |\mathrm{rv}(F_R)|$
- *Offline* Varianten suchen eine Reihenfolge $\mathcal{U}$, so dass VE mit $\mathcal{U}$ aufgerufen werden kann
  - Nächstes U_i mit dem kleinsten Zwischenergebnis wählen (wie online, nur bis zum Ende)
  - Nächstes U_i wählen, so dass am wenigsten neue Nachbarn zu den verbleibenden Zufallsvariablen hinzufügt werden (neue Nachbarn entstehen durch Multiplikation)

VE Algorithmus mit Online-Heuristik h

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

for $t = 1, \dots, m$ **do** — Evidenz behandeln
  while $\exists f \in F : T_t \in \mathrm{rv}(f)$ **do**
    $F \leftarrow F \setminus \{f\} \cup \{\mathrm{ABSORB}(f, T, \phi_t(T))\}$ ▹ Absorbiere $\phi_t(T_t)$ in F

while $\mathrm{rv}(F) \setminus \boldsymbol{S} \neq \emptyset$ **do** — Nichtanfrage-Variablen eliminieren
  $U \leftarrow \arg\min_R h(F)$ ▹ Wähle nächstes U zur Eliminierung
  while $\exists f_1, f_2 \in F : U \in \mathrm{rv}(f_1) \wedge \mathrm{rv}(f_2)$ **do**
    $F \leftarrow F \setminus \{f_1, f_2\} \cup \{\mathrm{MULTIPLY}(f_1, f_2)\}$ ▹ Multipliziere f_1, f_2 in F
  $F \leftarrow F \setminus \{f\} \cup \{\mathrm{SUM\text{-}OUT}(f, U)\}$ ▹ Summiere U aus F aus

while $\exists f_1, f_2 \in F$ **do** ▹ Multipliziere f_1, f_2 in F, bis $|F| = 1$
  $F \leftarrow F \setminus \{f_1, f_2\} \cup \{\mathrm{MULTIPLY}(f_1, f_2)\}$

Normalisiere die Potentiale in $f \in F$ — Normalisieren

return f

Laufzeitkomplexität

- *Informell*
  Worst-case-Größe eines Zwischenergebnisses *mal* Anzahl an Eliminierungen
- Dekompositionsbaum (*decomposition tree*, *dtree*)
  - Datenstruktur gebraucht zur formalen Bestimmung der Komplexität
  - Repräsentiert einen VE Durchlauf, z.B., für die leere Anfrage $P(.)$
  - Azyklischer Baum
    - Für BNs in der Regel als Binärbaum definiert
  - Blätter: Faktoren aus dem Eingabemodell
  - Innere Knoten: Zwischenergebnisse
    - Wurzel: Endresultat
  - Kanten: Wenn (Faktoren multipliziert und dann) eine Zufallsvariable von einem Zwischenergebnis / Faktor eliminiert wird, was zu einem neuen Zwischenergebnis (innerer Knoten) führt, dann gibt es eine Kante von den Zwischenergebnissen / Faktoren vor der Multiplikation zum inneren Knoten

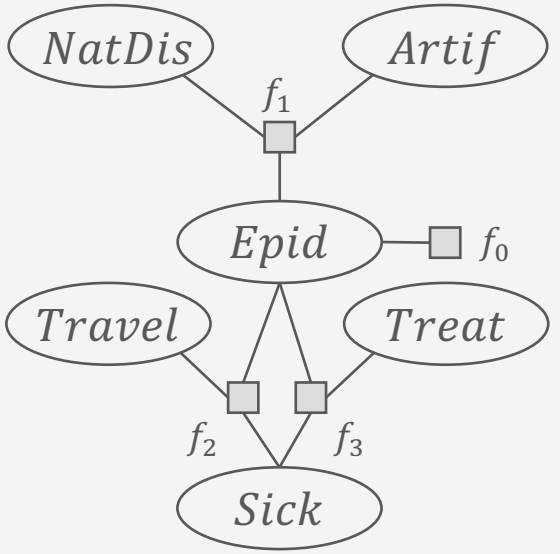
Mehr Infos: Darwiche (2009), Kap. 8.3, 9.5

Beispiel: Berechnung von $P(.)$

$$P(.) = \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \underbrace{\sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a)}_{\phi_1'(e,n)} \sum_{s \in \mathrm{Val}(S)} \underbrace{\sum_{tr \in \mathrm{Val}(Tr)} \phi_2(tr,e,s)}_{\phi_2'(e,s)} \underbrace{\sum_{tt \in \mathrm{Val}(Tt)} \phi_3(e,s,tt)}_{\phi_3'(e,s)}$$

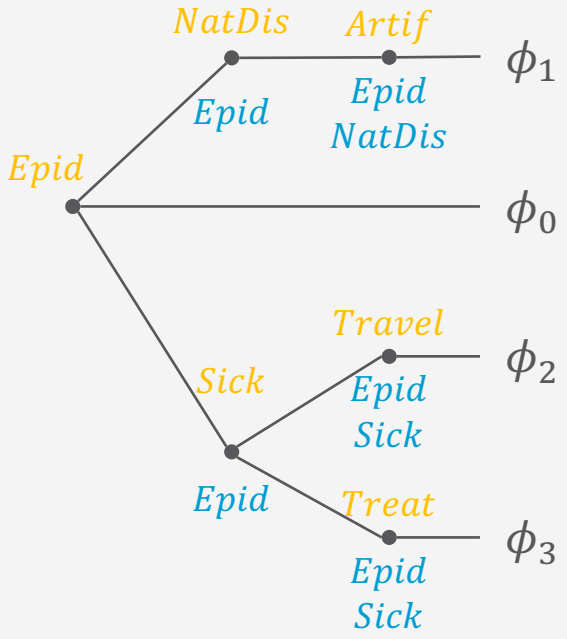
$$= \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \underbrace{\sum_{n \in \mathrm{Val}(N)} \phi_1'(e,n)}_{\phi_1''(e)} \underbrace{\sum_{s \in \mathrm{Val}(S)} \phi_2'(e,s)\phi_3'(e,s)}_{\phi_{23}(e,s) \rightarrow \phi_{23}'(e)}$$

$$= \underbrace{\sum_{e \in \mathrm{Val}(E)} \phi_0(e)\phi_1''(e)\phi_{23}'(e)}_{\phi(e) \rightarrow \phi'(.)}$$



Beispiel: Dtree – Bottom-up Aufbau als VE Repräsentation

$$P(.) = \sum_{e \in \mathrm{Val}(E)} \phi_0(e) \sum_{n \in \mathrm{Val}(N)} \sum_{a \in \mathrm{Val}(A)} \phi_1(e,n,a) \sum_{tr \in \mathrm{Val}(Tr)} \phi_2(tr,e,s) \sum_{s \in \mathrm{Val}(S)} \sum_{tt \in \mathrm{Val}(Tt)} \phi_3(e,s,tt)$$

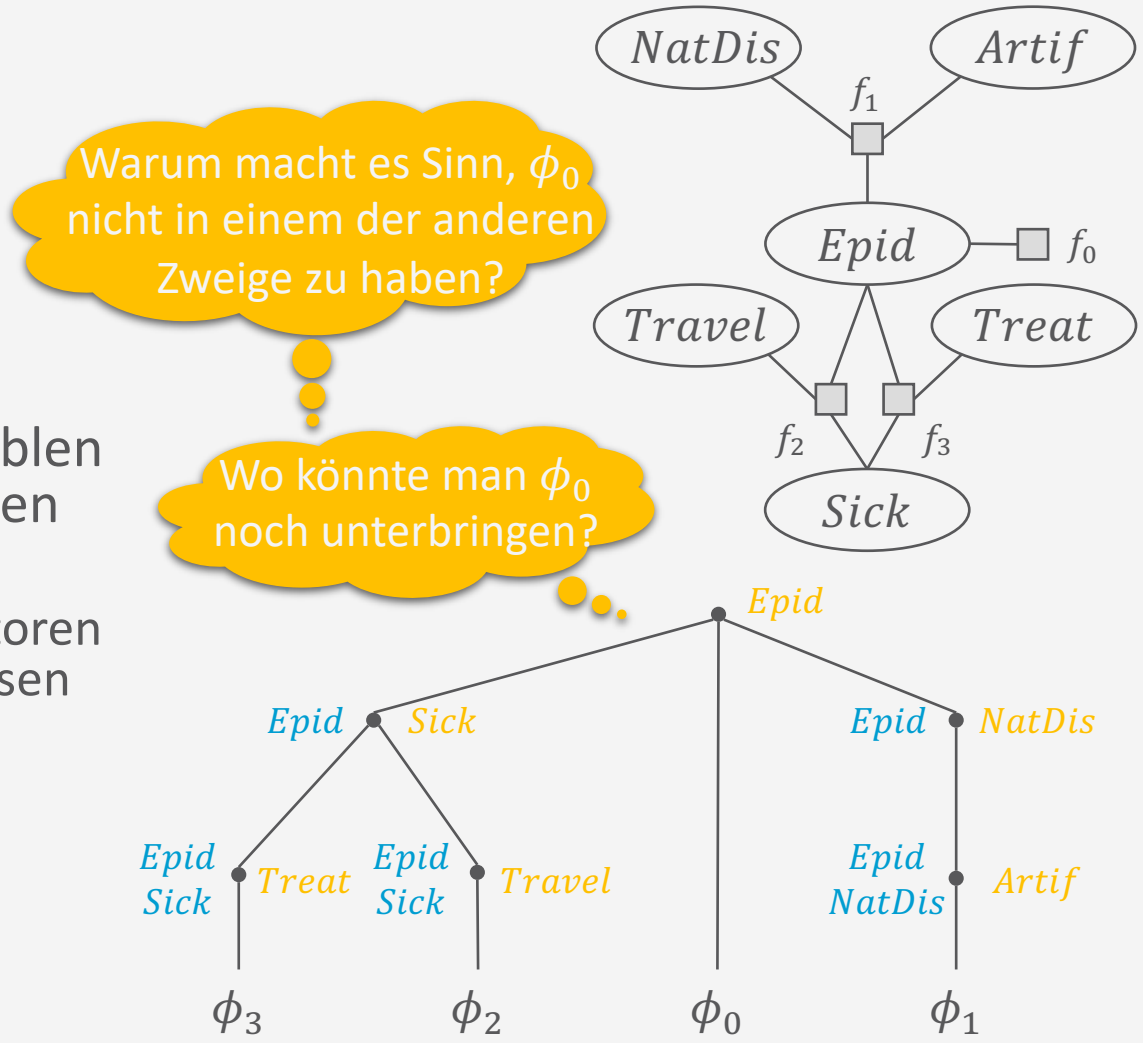


Berechnungen aus unterschiedlichen Zweigen können parallel ausgeführt werden, da sie unabhängig voneinander sind

$E \triangleq Epid, N \triangleq NatDis, A \triangleq Artif, S \triangleq Sick, Tr \triangleq Travel, Tt \triangleq Treat$

Beispiel: Dtree – Top-down Interpretation

- Zu Beginn: Wurzelknoten mit Eingabemodell $F = \{f_i\}_{i=1}^{n}$ als momentanes Modell F'
- Finde rekursiv Partitionen des momentanen Modells F' am Knoten k, so dass gilt
  - Jede Partition $F_i \subseteq F'$ enthält so viele Zufallsvariablen $\boldsymbol{U}_i$ wie möglich, welche nicht in anderen Partitionen vorkommen
    - Dann kann $\boldsymbol{U}_i$ eliminiert werden, ohne dass die Faktoren der anderen Partitionen berücksichtigt werden müssen
  - Füge für jede Partition einen Kindknoten i zu k mit F_i als momentanem Modell F' hinzu
- So lange bis jedes momentane Modell F' an einem Knoten nur noch einen Faktor enthält



Cutset, Kontext, Cluster

- Cutset
  - Was an diesem Knoten eliminiert wird (*cut from the model*)

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

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

$\text{rv}(T)$ gibt die Zufallsvariablen zurück, die in den Faktoren dieses (Unter-) Baums vorkommen

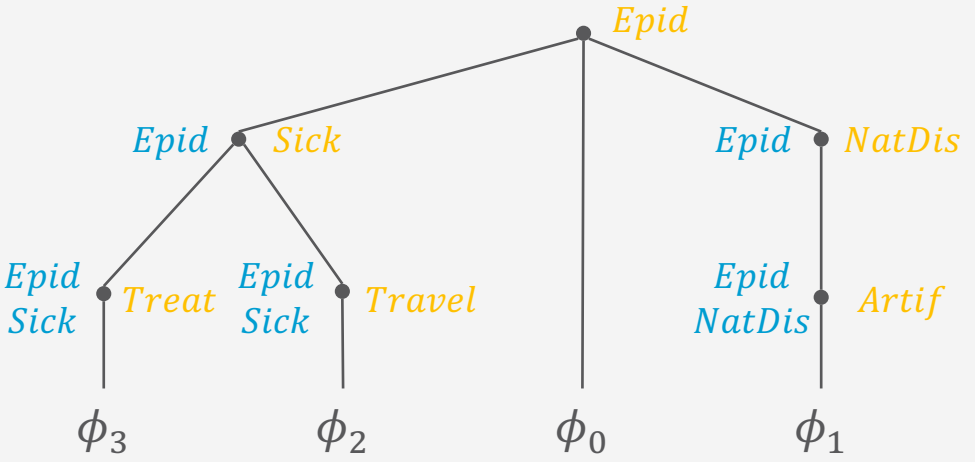
- Kontext
  - Was bei der Eliminierung gesetzt ist (sonst noch vorkommt)

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

- Cluster
  - Cutset und Kontext zusammen

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

    - Wenn T Blatt ist, dann $\text{cluster}(T) = \text{context}(T) = \text{rv}(f)$

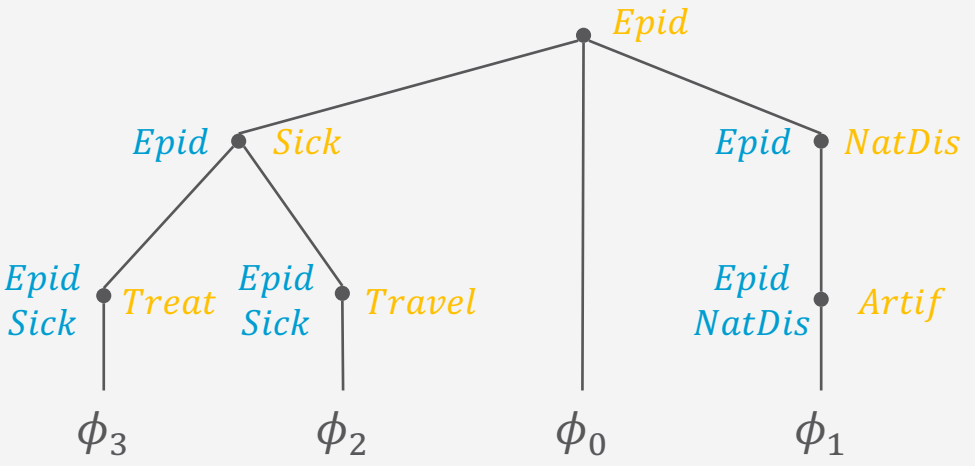


Cutset, Kontext, Cluster

- Größtes Cluster in Baum T
  = Baumweite (*tree width*) w

$$w = \max_{T_i \in \mathrm{Desc}(T)} |\mathrm{cluster}(T_i)|$$

- Induziert eine Worst-case-Faktorgröße
  - Cluster gibt an, welche Zufallsvariablen an einer Eliminierung beteiligt sind
    - Kommen zusammen in einem Faktor vor
  - Größtes Cluster → größte Anzahl an Argumenten, die zusammen in einem Faktor vorkommen
  - Beispiel:
    - $w = 3$, Worst-case-Faktorgröße $2^w = 2^3$



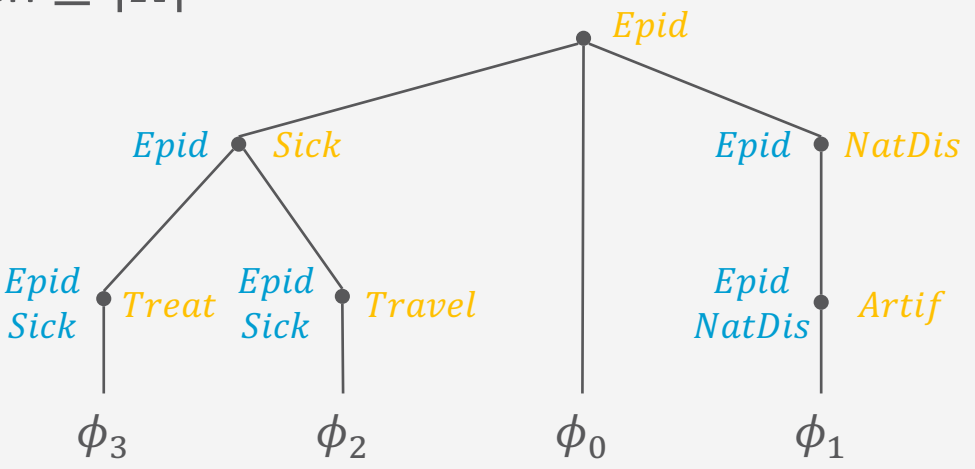
Zurück zur Laufzeitkomplexität

- *Informell*
  Worst-case-Größe eines Zwischenergebnisses *mal* Anzahl an Eliminierungen
- Dekompositionsbaum (*decomposition tree*, *dtree*)
  - Baumweite w = Worst-case-Größe eines Zwischenergebnisses / Faktors
  - Anzahl an inneren Knoten n_T = Anzahl an Eliminierungen $\leq |\boldsymbol{R}|$

- *Formal*:
  Laufzeitkomplexität von VE

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

  - $r = \max_{R \in \boldsymbol{R}} |\mathrm{Val}(R)|$



Laufzeitkomplexität Vergleich

- Laufzeitkomplexität von VE: $O(n_T \cdot r^w)$
  - w von unten begrenzt durch m:

$$w \geq m = \max_{f \in F} |\mathrm{rv}(f)|$$

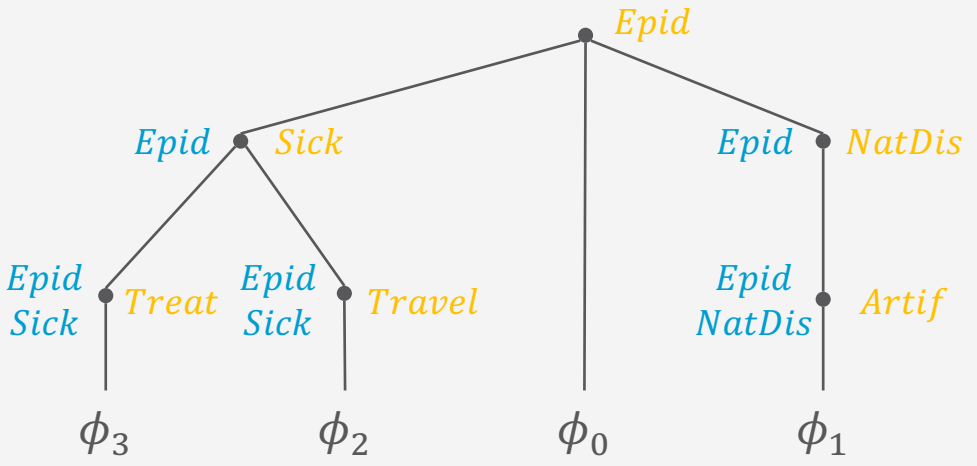
    - Wenn man ein Modell lernt, Faktoren mit vielen Argumenten vermeiden / Anzahl an Argumenten bzw. $\deg(F)$ im FG bzw. MN beschränken
  - Vergleich Inferenz über die vollständige gemeinsame Verteilung P_F: $O(r^{n_T})$ bei $n_T = n = |\boldsymbol{R}|$
  - Vergleich Speicherkomplexität
    - Faktormodell: $O(n' \cdot r^m), n' = |F|$
    - BN: $O(n \cdot r^m), n = |\boldsymbol{R}|$
    - $m = \max_{f \in F} |\mathrm{rv}(f)|$ die maximale Anzahl an Argumenten pro Faktor im Faktormodell

Komplexität und Traktabilität (*tractability*)

- Laufzeitkomplexität von VE: $O(n_T \cdot r^w)$
  - w von unten begrenzt durch m:
$$w \geq m = \max_{f \in F} |\mathrm{rv}(f)|$$
    - Wenn man ein Modell lernt, Faktoren mit vielen Argumenten vermeiden / Anzahl an Argumenten bzw. $\deg(F)$ im FG bzw. MN beschränken
  - Vergleich Inferenz über die vollständige gemeinsame Verteilung P_F: $O(r^{n_T})$ bei $n_T = n = |\boldsymbol{R}|$
- Anfragebeantwortungsproblem ist traktabel (*tractable*) WENN
  - es durch einen effizienten Algorithmus gelöst wird, der in polynomieller Zeit in Bezug auf die Anzahl der Zufallsvariablen läuft
- Anfragebeantwortungsproblem im Allgemeinen ist nicht traktabel (*intractable*)
  - Keine Garantien, dass $w \ll n_T$
  - Es sei denn, man macht Annahmen, z.B. über die Modellstruktur (wie Polytree BNs)

VE, Dtrees und Unabhängigkeiten

- Globale Unabhängigkeiten ermöglichen die Faktorisierung der gemeinsamen vollständigen Verteilung, die VE dann wiederum für effiziente Anfragenbeantwortung nutzt
  - Globale Unabhängigkeiten im Beispielmodell:
    - $Treat, Travel, Sick \perp NatDis, Artif \mid Epid$
    - $Treat \perp Travel \mid \{Epid, Sick\}$
- Unabhängigkeiten auch im Dtree wiederzufinden
  - Beispiel
    - Rechter Zweig der Wurzel beinhaltet $NatDis, Artif$
    - Linker Zweig der Wurzel beinhaltet $Treat, Travel, Sick$
    - In beiden Zweigen $Epid$ im Kontext
      → Zweige damit unabhängig voneinander
      - Parallele Berechnung von daher korrekt



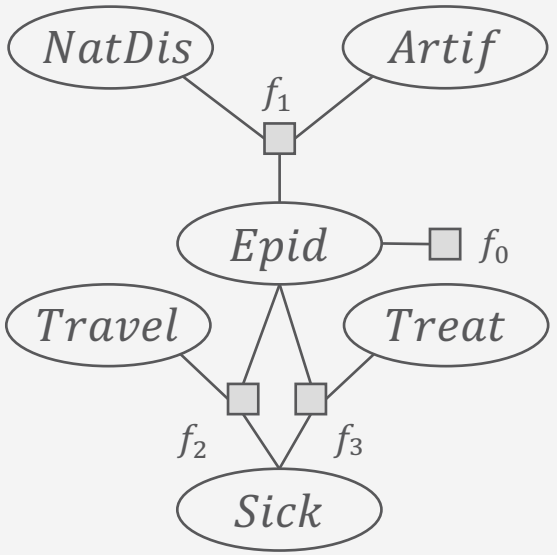
Implementierung von VE mit Heuristik

- Implementierung von Nima Taghipour
  - Als Teil seiner Doktorarbeit zu *Lifted VE*
    - Früher verfügbar hier: https://dtai.cs.kuleuven.be/software/gcfove
  - Verfügbar als Teil von einer *Lifted JT* Implementierung
    - Verfügbar hier: https://www.ifis.uni-luebeck.de/index.php?id=590&L=4
  - Eingabeformat basierend auf BLOG:
    - Verfügbar hier: https://bayesianlogic.github.io
    - Probabilistische Modellierungssprache
      - Kontinuierliche Modellierung, Sampling-Algorithmen
    - Einige Versionen von BLOG beinhalten auch VE Implementierungen
    - Das Eingabeformat erlaubt allerdings keine einfache Definition von Faktoren
    - Definition von CPTs über *if-then-else*-Konstrukte möglich

Werbeblock (wiederholt):
Vorlesung *„Relational Inference and Online Decision Making“*

Implementierung: BLOG Input

- Komponenten
  - Zufallsvariablen
    - Boolesche Domäne gegeben, auch benutzerdefinierte Werte möglich
  - Faktoren
  - Anfragen, Beobachtungen
- Liste von Potentialen
  - Bei Booleschen Werten
    - Beginn: alle $true$
    - Ende: alle $false$
    - Wenn man die Zuweisungen als Binärzahlen interpretiert, dann geht man absteigend durch sie durch



BLOG file

```
random Boolean Epid;
random Boolean Sick;
random Boolean Travel;
...
factor MultiArrayPotential[[1, 50]] (Epid);
factor MultiArrayPotential
        [[2,7,8,28,6,5,24,20]]
        (Travel, Epid, Sick);
...
query Travel; // Anfrage

obs Sick=true; // Beobachtung
```

Implementierung: Ausgabe

- Aufruf: `./runblog examples/epid.blog -e ve.VarElimEngine`

  Skript für Programmaufruf | Eingabedatei | Inferenzmaschine

- Ausgabe:

```
Using fixed random seed for repeatability.
Parsing from: examples/epid.blog
.....................................
.................. VE ..................
Constructing inference engine of class ve.VarElimEngine
-----------------------------------------------------
Trial 0:

**TIME**2569628
======== Query Results =========
Distribution of values for Travel
0.75 false
0.25 true
```

(Wichtig für Sampling)

(Wichtig für Sampling)

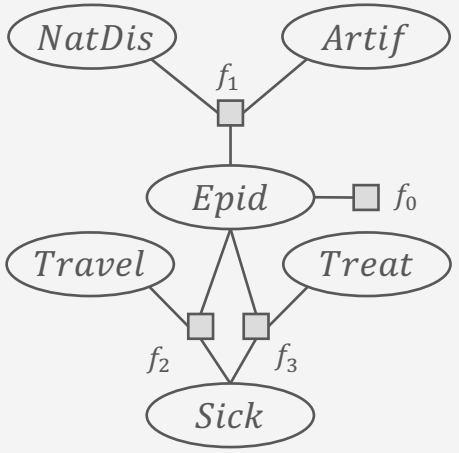
Laufzeit in Nanosekunden

Anfrageergebnisse

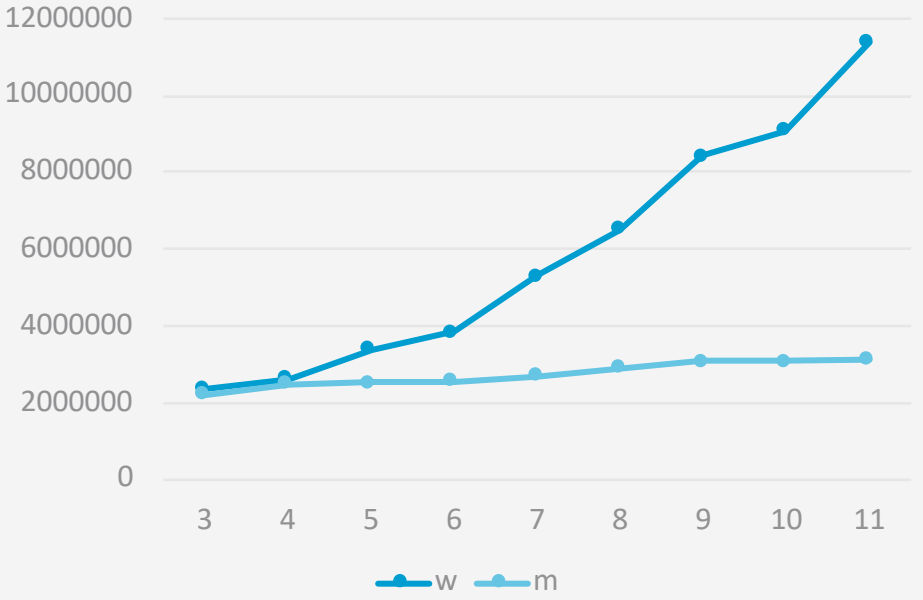
Dahinter folgen noch ein paar Zahlen für die Statistik, die vor allem für die *Lifting* Varianten wichtig sind.

Implementierung: Laufzeiten

- Laufzeitkomplexität von VE: $O(n_T \cdot r^w)$
- Verhalten bei steigendem
  - w: Immer mehr neue Zufallsvariablen zu Faktoren hinzufügen
  - n_T: Immer mehr Faktoren an $Epid$ mit zwei neuen Zufallsvariablen anfügen



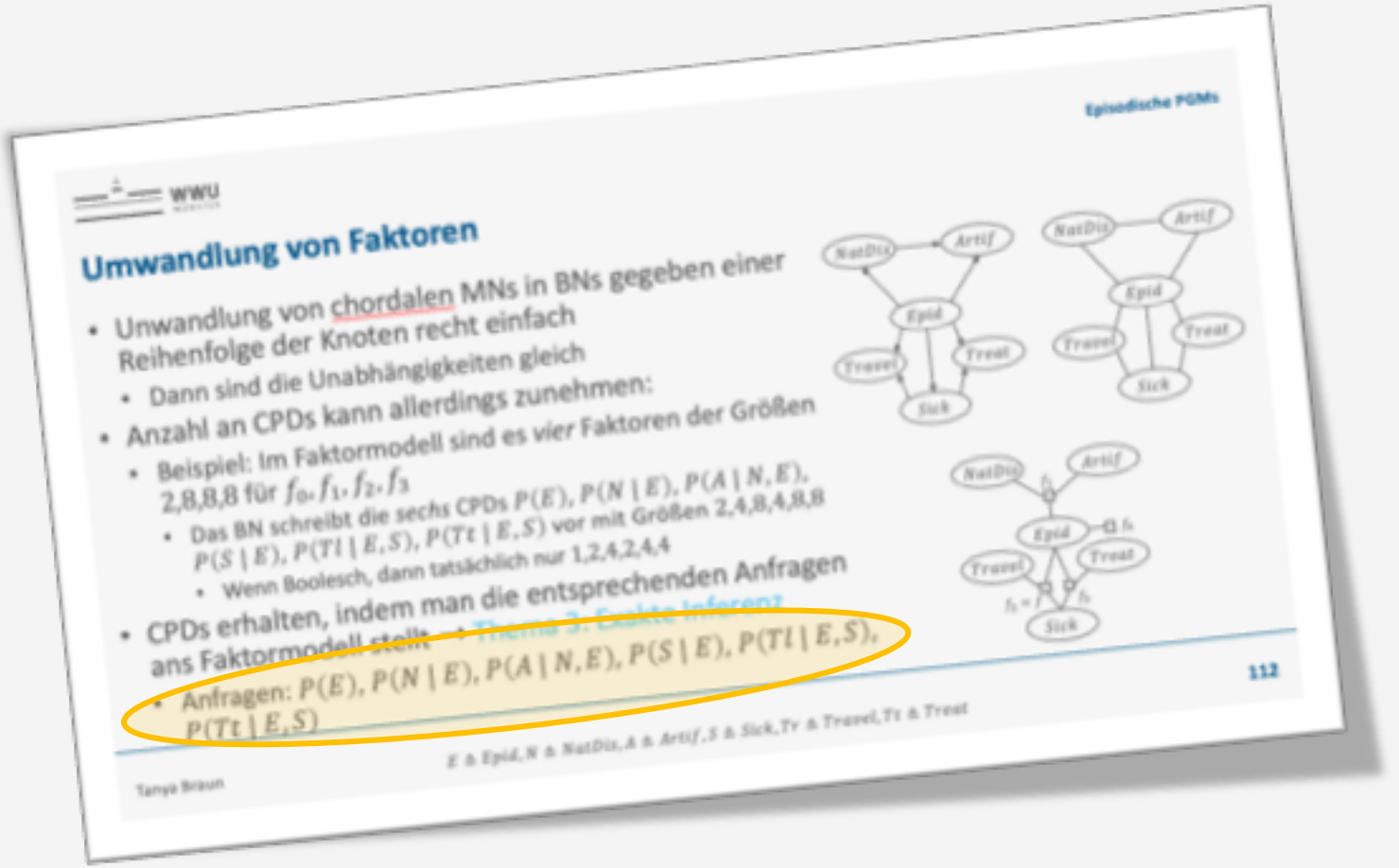
- Laufzeiten:
  - Exponentieller vs. linearer Anstieg sichtbar:



m = Anzahl Dreier-Cliquen: $n_T = m \cdot 2$

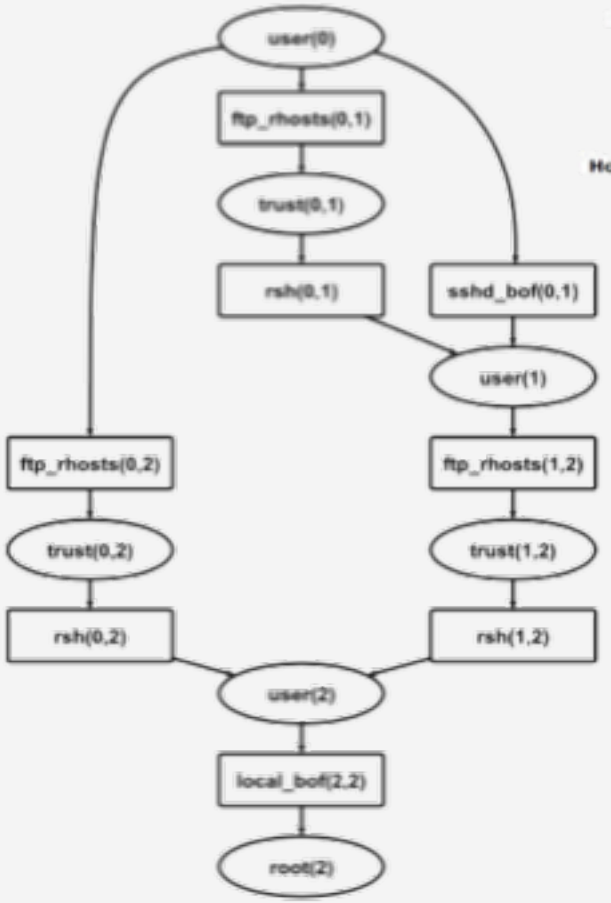
Anwendung: VE für die Umwandlung von MNs in BNs

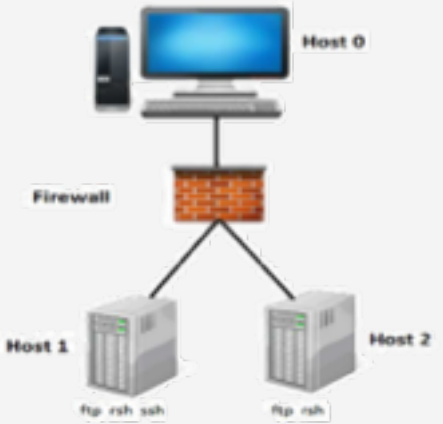
- Bedingte Anfragen an Faktormodell F um CPDs für BN B zu erhalten
  - Beispiel: $P(Epid)$
    - $\mathrm{VE}(F, \{Epid\}, \emptyset)$
      - Alle Zufallsvariablen außer $Epid$ eliminieren, verbleibenden Faktor normalisieren
      - Ergebnis als Aprior-Verteilung in BN übernehmen

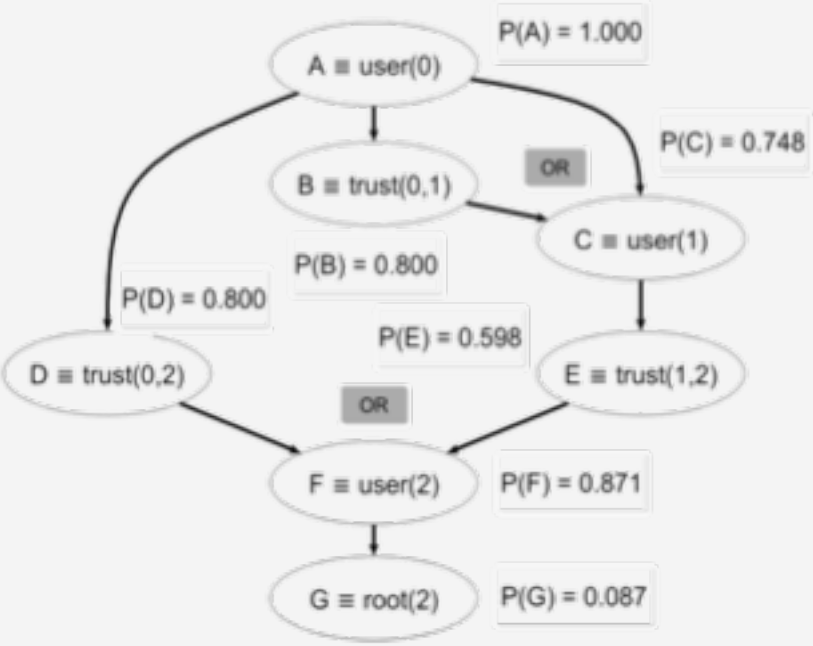


Anwendung in *Bayesian Attack Graphs*

- Modellierung zur Analyse von Sicherheitsrisiken in Netzwerken
  - *Logical Attack Graphs*
    - Angriffspunkte/pfade bzw. Angriffe
    - Zusammenfassung von Nutzergruppen in Knoten
  - Erweitert um Unsicherheiten
    - (Bedingte) Wahrscheinlichkeiten über Kompromittierung von Angriffspunkten
- Anfragen an Wahrscheinlichkeits-verteilungen zu jedem Knoten







Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C. Lupu: Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. In: *IEEE Transactions on dependable and Secure Computing*, 2017.
Abbildungen ebenda.

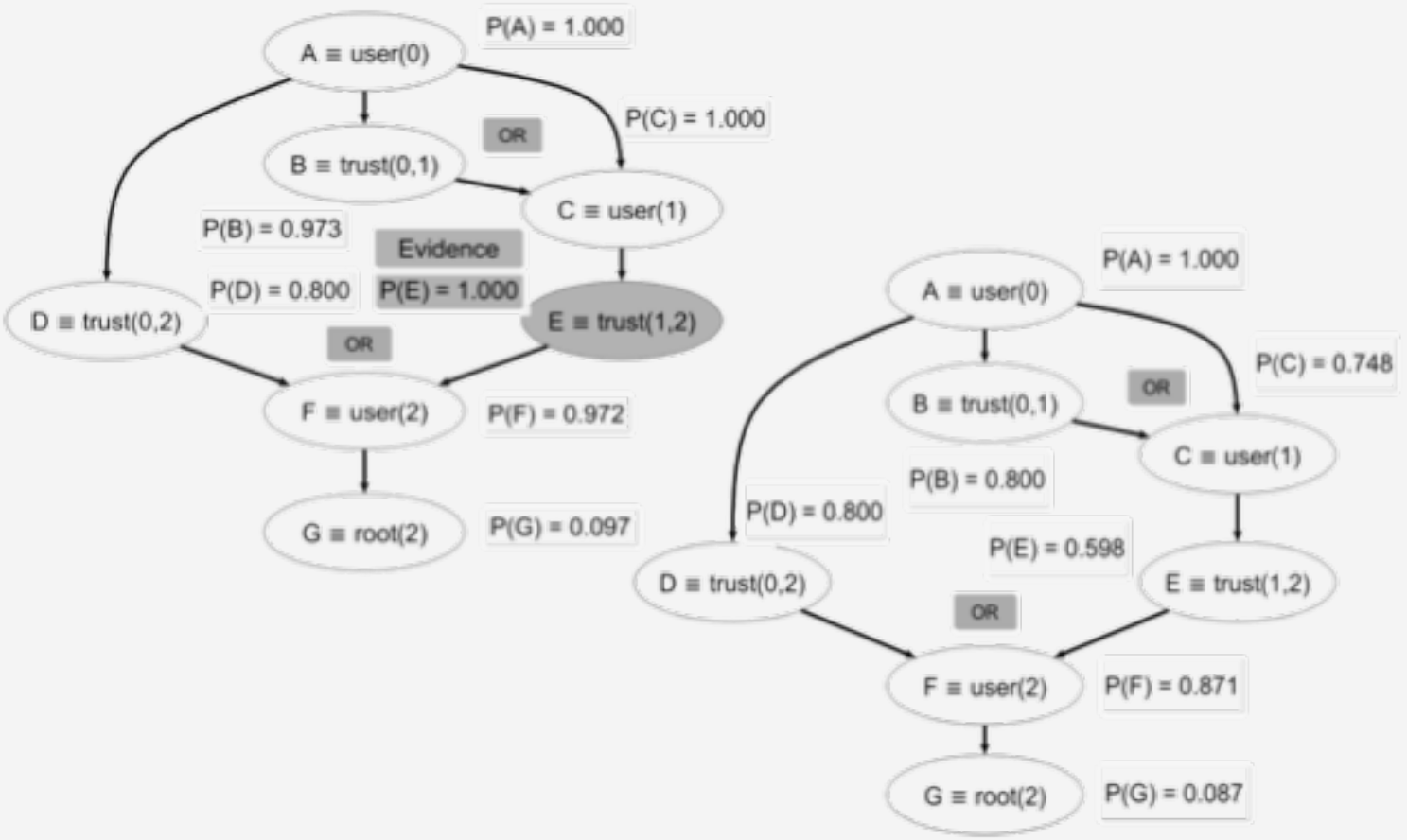
Anwendung: *Bayesian Attack Graphs*

- Anfragen an Wahrscheinlichkeits-verteilungen zu jedem Knoten
  - Erfordert die Eliminierung aller jeweils anderen Knoten
- Anfragen an bedingte Wahrscheinlichkeitsverteilungen bei unterschiedlicher Evidenz

→ Viele doppelte Rechnungen

→ Potential für Verbesserungen

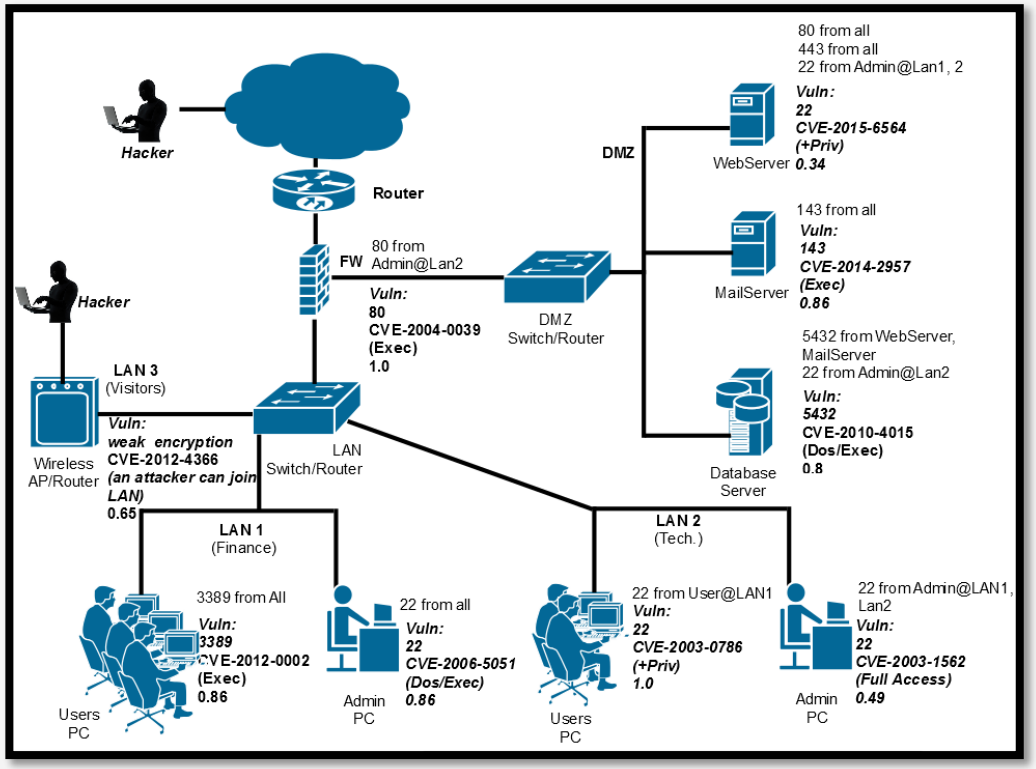
  - *Besonders, wenn die Netze größer werden*

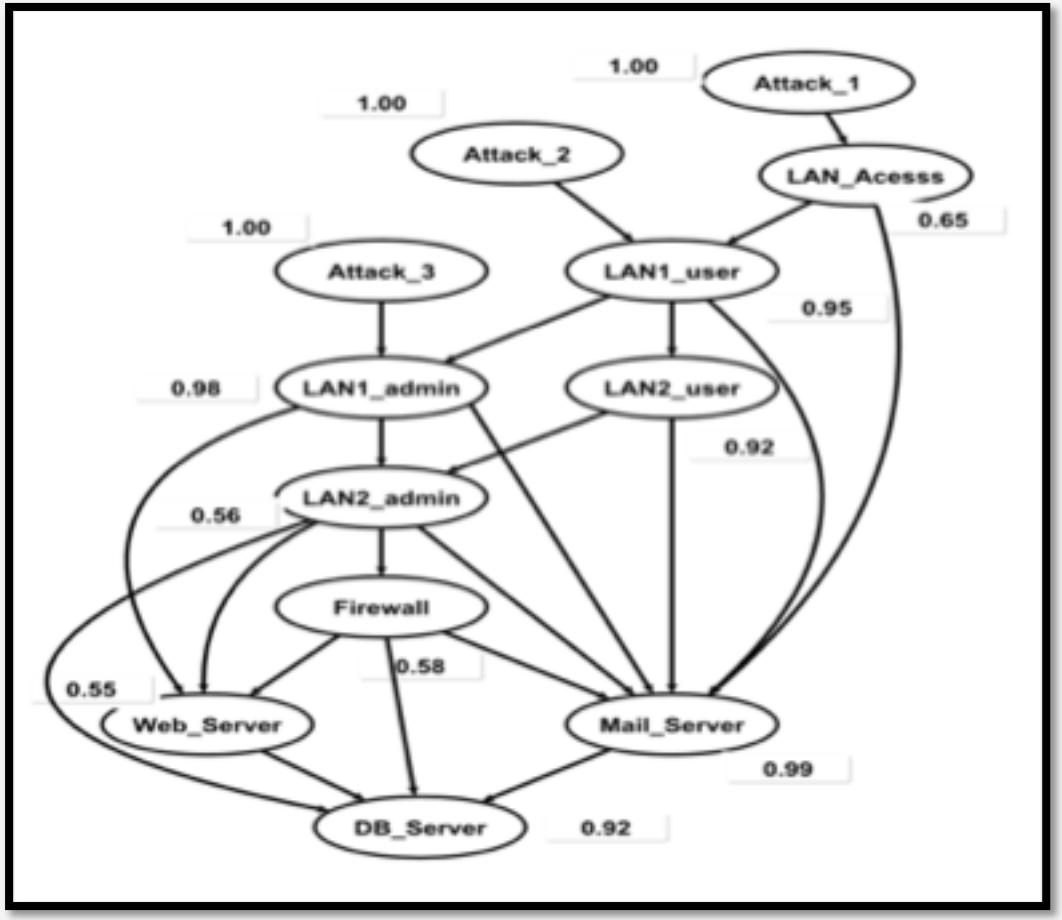


Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C. Lupu: Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. In: *IEEE Transactions on dependable and Secure Computing*, 2017.
Abbildungen ebenda.

Anwendung: *Bayesian Attack Graphs*

- Typisches Netzwerk eines klein- bis mittelständischen Unternehmens





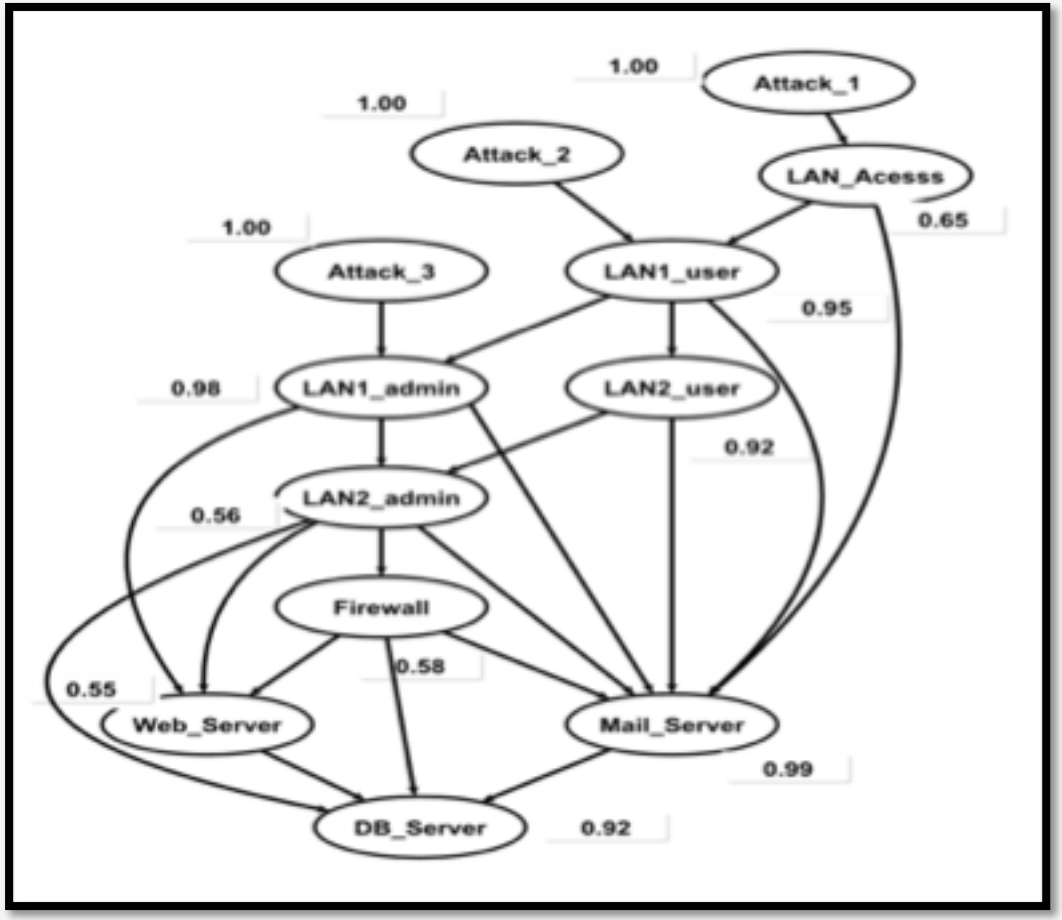
Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C. Lupu: Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. In: *IEEE Transactions on dependable and Secure Computing*, 2017.
Abbildungen ebenda.

Anwendung: *Bayesian Attack Graphs*

- Anfragen: $P(A_1), P(A_2), P(A_3), P(L_{ac}), P(L_{1u}), P(L_{2u}), P(L_{1a}), P(L_{2a}), P(F), P(S_w), P(S_m), P(S_{db})$
  - Von einer Anfrage zur nächsten ändert sich eine Summe bzw. jeweils eine fällt raus aus $P(.)$:

$$\sum_{s_{db}} \sum_{s_m} \sum_{s_w} \sum_{f} \sum_{l_{2a}} \sum_{l_{1a}} \sum_{l_{2u}} \sum_{l_{1u}} \sum_{l_{ac}} \sum_{a_3} \sum_{a_2} \sum_{a_1} P_B$$

  - Gute Eliminationsreihenfolgen ändern sich nur minimal zwischen benachbarten Knoten
  - Ergebnisse von aufwendigen Aussummierungen möchte man wiederverwenden
  - Problem der Mehrfachanfragen effizient lösen



Luis Muñoz-González, Daniele Sgandurra, Martín Barrère, and Emil C. Lupu: Exact Inference Techniques for the Analysis of Bayesian Attack Graphs. In: *IEEE Transactions on dependable and Secure Computing*, 2017.
Abbildungen ebenda.

Zwischenzusammenfassung

- VE Beispiel mit und ohne Evidenz
- VE Algorithmus
  - VE Operatoren: SUM-OUT, MULTIPLY, ABSORB
  - VE in BNs → Faktoren
  - Reihenfolge der Eliminierungen
    - Heuristik: online, offline
- Laufzeitkomplexität $O(n_T \cdot r^w)$
  - Dtrees als VE-Repräsentation
    - Cutset, Kontext, Cluster
  - Baumweite w als maximales Cluster
- Implementierung
- Problem: Mehrfachanfragen ineffizient

Überblick: 3. Exakte Inferenz in episodischen PGMs

A. *Einzelanfragen: Variableneliminierung (VE)*
   - Algorithmus, Operatoren für Wahrscheinlichkeitsanfragen
   - Dekompositionsbäume, Komplexität

B. Multi-Anfragen: Junction Tree (Cliquen-Bäume) Algorithmus (JT)
   - Cliquen, Junction Tree als Hilfsstruktur, Vorverarbeitung und Anfragebeantwortung
   - Zusammenhang mit VE, Komplexität
   - *Geschichtsstunde: Pearl's Probability Propagation (PP)* auf Polytree BNs

C. *Inferenzproblem Zustandsanfragen*
   - Ausprägungen: Most probable explanation (MPE) / maximum a posteriori Anfragen (MAP)
   - Semantik: Ausmaximieren statt aussummieren; max-out Operator in VE
   - Auswirkungen auf die Komplexität