

# Lifted First-Order Probabilistic Inference



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Matthias Winges



# Agenda

- i. Einführung/Problembeschreibung
- ii. Lösungsvorschläge
- iii. First-Order Probabilistic Models (FOPM)
- iv. Inference Problem
- v. First-Order Variable Elimination (FOVE)
- vi. Fazit & Ausblick



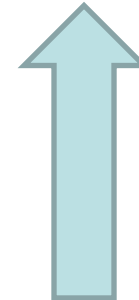
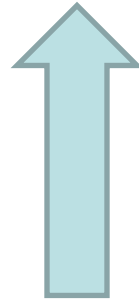
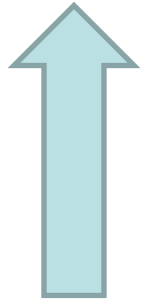
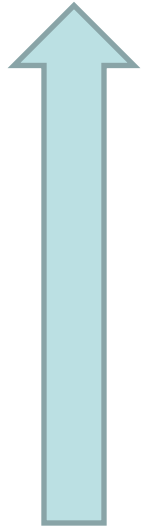
# Einführung (1)

▪ Lifted

First-Order

Probabilistic

Inference



Vgl. frühere  
Vorträge

Wahrscheinlichkeiten  
spielen eine Rolle

Schließen

Schließen findet direkt auf first-order Ebene statt; Variablen werden nur instanziiert, wenn dies wirklich notwendig ist

# Einführung (2)

- Schließen unter Wahrscheinlichkeiten ist schwieriger:
  - Es müssen alle Informationen genutzt werden und neue können alte Aussagen ändern
  - Gefahr Dinge doppelt zu zählen, wenn z.B. Information über Person A betrachtet wird und dann eine Information über Gruppe von Personen, in der A enthalten ist
  - Domänengröße kann Wahrscheinlichkeiten beeinflussen (z.B. Wahrscheinlichkeit, dass eine Person ein bestimmtes Verbrechen begangen hat)

# Einführung (3)

- *Probabilistic inference algorithms* → deklarative Aussagen
- Oft bessere Repräsentation durch first-order Beschreibung

# Einführung (3)

- *Probabilistic inference algorithms* → deklarative Aussagen
- Oft bessere Repräsentation durch first-order Beschreibung

- Beispiel:

- Aussagenlogik

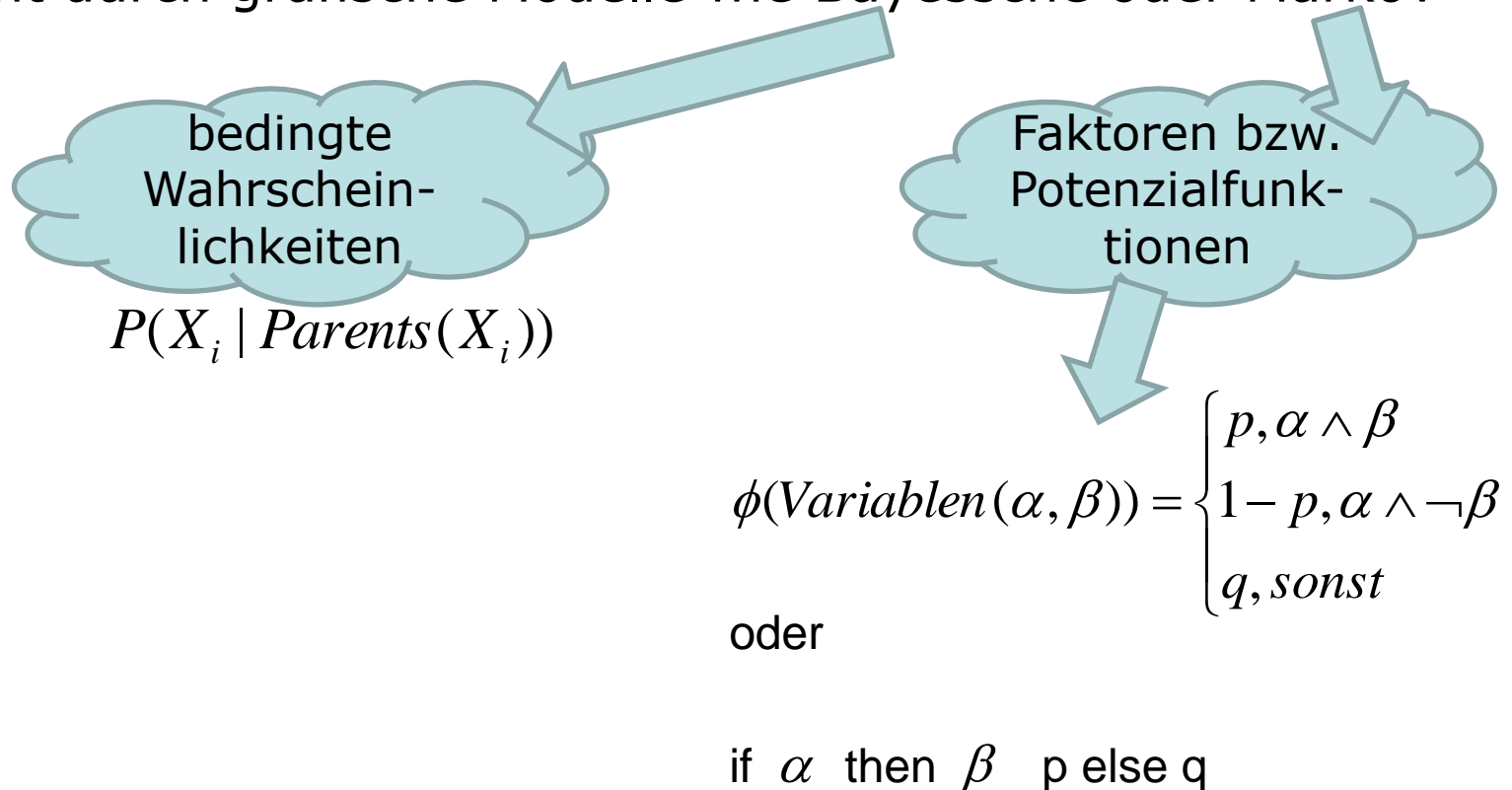
if *epidemic* then *sick*(john) 0.7  
if *sick*(john) then *death*(john) 0.4  
if *epidemic* then *sick*(marry) 0.7  
if *sick*(marry) then *death*(marry) 0.4  
...

- First-order logic

if *epidemic* then *sick* 0.7  
if *sick* then *death* 0.4

# Einführung (4)

- *Probabilistic inference algorithms* kommen aus der KI
- Bekannt durch grafische Modelle wie Bayessche oder Markov Netze



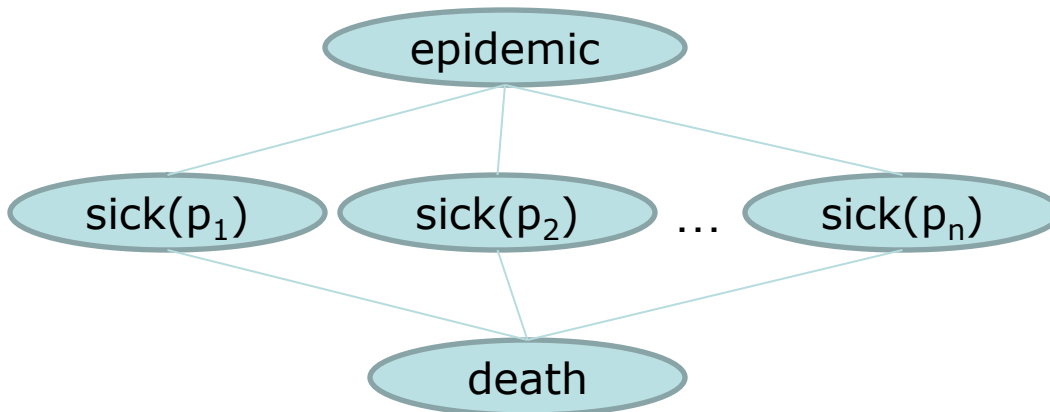
# Einführung (5)

- **Problem** bisheriger Algorithmen:  
first-order Repräsentation → Schließen (Inference) aber auf  
Basis aussagenlogischer Repräsentation
  - „probabilistic inference is NP-hard [...] research should be  
directed away from the search for a general, efficient  
probabilistic inference algorithm...” (G. F. Cooper)
  
- Hier dargestellter **FOVE**-Algorithmus:
  - Beschleunigung von Inference durch Ausnutzen der first-order  
Darstellung:
    - Generalisierung von **variable elimination**

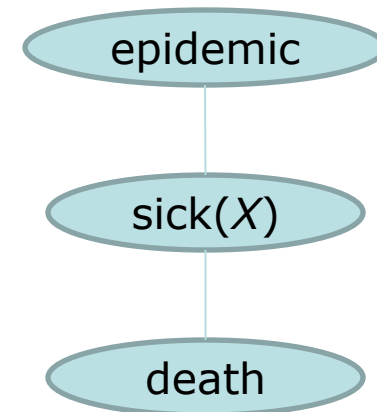


# Einführung (6)

- Beispiel, um Beschleunigung zu illustrieren:
  - Beispiel auf Folie 3 folgend: Abfrage, ob irgendeine Person stirbt



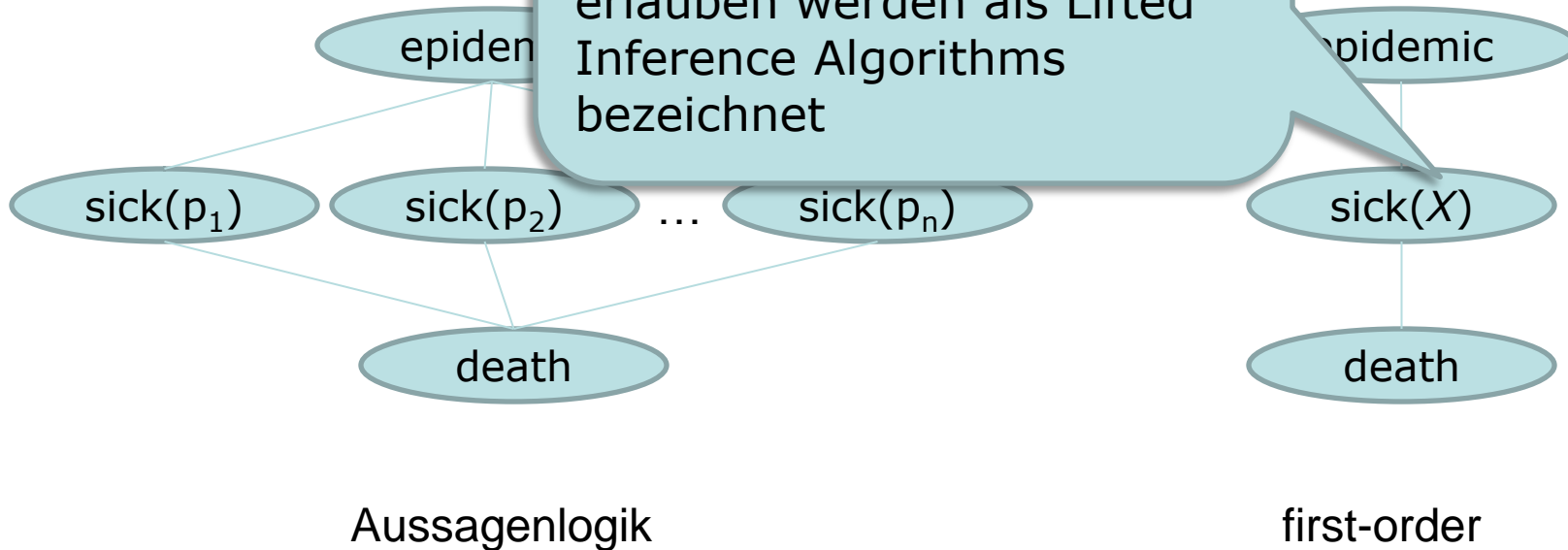
Aussagenlogik



first-order

# Einführung (6)

- Beispiel, um Beschleunigung zu illustrieren:
  - Beispiel auf Folie 3: X muss nicht n-mal instanziiert werden  
→ Algorithmen, welche dies erlauben werden als Lifted Inference Algorithms bezeichnet



# Lösungsvorschläge (1)

- Erste Lösung: Algorithmus von Poole (2003)
  - **Inversion Elimination** von Zufallsvariablen
  - Parametrisierte Zufallsvariablen stehen für alle ihre Instanzen
  - Aber: Nur gleich parametrisierte Zufallsvariablen können eliminiert werden
    - Beispiel: *product(X,Y)* kann eliminiert werden; *company(X)* nicht, da *Y* fehlt
- Lösung von Braz, Amir, Roth:
  - Erweiterung um:
    - Counting Elimination
    - Inversion (bzw. Partial Elimination)



## First-order variable elimination (FOVE)

Versuch des Lückenschlusses zwischen logischem und wahrscheinlichkeitsbasiertem Schließen

# Lösungsvorschläge (1)

- Erste Lösung: Algorithmus von Poole  
→ **Inversion Elimination** von Zufalls
- Parametrisierte Zufallsvariablen stehen für
- Aber: Nur gleich parametrisierte Zufalls  
werden
  - Beispiel:  $product(X, Y)$  kann eliminiert  
da  $Y$  fehlt

überlegene Performanz  
gegenüber exaktem  
aussagenlogischen  
Schließen  
(„propositional exact  
inference“)

- Lösung von Braz, Amir, Roth:  
→ Erweiterung um:
  - Counting Elimination
  - Inversion (bzw. Partial Elimination)

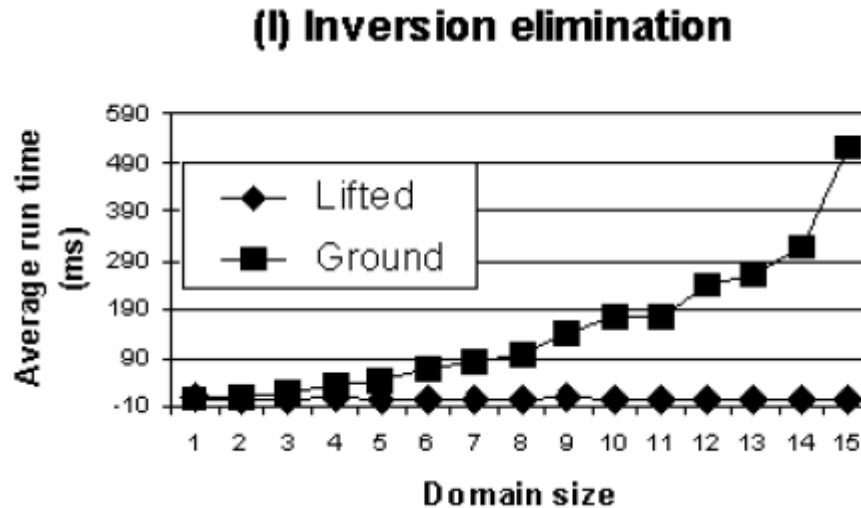


## First-order variable elimination (FOVE)

Versuch des Lückenschlusses zwischen logischem und wahrscheinlichkeitsbasiertem Schließen

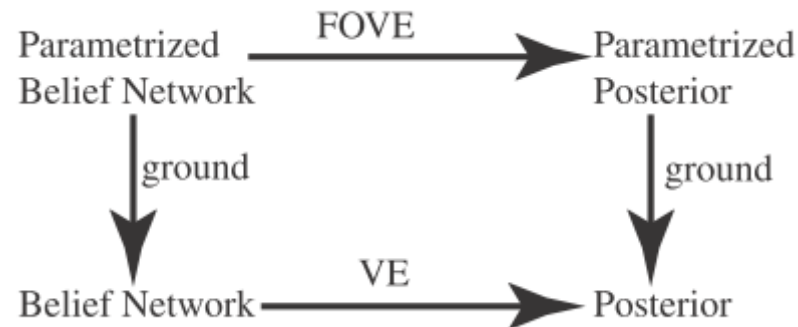
# Lösungsvorschläge (2)

- Inversion Elimination auf Abfrage ***P(death)*** auf  
{epidemic 0,55; if epidemic then sick(X) 0,7 else 0,01;  
if sick(X) then death 0,55}



# Lösungsvorschläge (3)

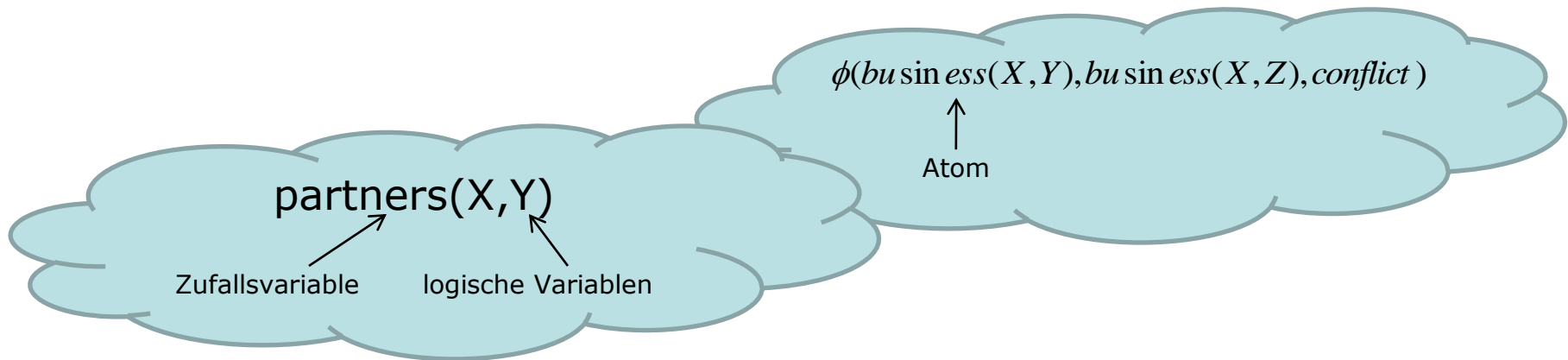
- Poole
  - Oft ist man an groundings nicht interessiert und möchte Gruppen (z.B. von Personen) behandeln
  - First-Order Variable Elimination (FOVE)



# First-Order Probabilistic Models (FOPM) (1)

## ▪ Begriffe:

- Menge von Potenzialfkt./Faktoren, welche über parametrisierte Zufallsvariablen definiert sind → sog. **parfactors**
- Parametrisierte Zufallsvariable → **atom** (steht aber für ganze Klasse von Zufallsvariablen)
- Eine instanziierte Zufallsvariable → **ground atom**
- Parameter → **logical variable**



# First-Order Probabilistic Models (FOPM) (2)

- Parfactor  $g \rightarrow (\phi_g, A_g, C_g)$

Potenzialfunktion  
über Atomen  $A_g$

Instanziiert durch alle Substitutionen  
der logischen Variablen, welche  
Bedingungen  $C_g$  genügen

- Bedingung  $C = (F, V)$

Bedingung auf Menge logischer Variablen,  
die instanziiert werden



# First-Order Probabilistic Models (FOPM) (3)

- Beispiel:  $(\phi, (p(X), q(X, Y)), (X \neq a, \{X, Y\}))$



Substitution aller  
 $X, Y$ , für die Bedingung  
 $C$  erfüllt ist



Bedingung, dass  $X$   
nicht  $a$  sein darf

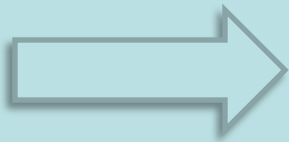


- Potenzialfunktionen können verschieden spezifiziert sein
  - Z.B. als logische Formeln  
 $0,7 : epidemic(D) \Rightarrow sick(P, D)$  steht für  $\Phi(epidemic(D), sick(P, D))$  mit Potenzial  $0,7$  für wahr
- Projektion von Bedingung  $C$  auf logische Variablen  $L$ :  $C|_L$  (wie etwa  $X \neq a$ )

# First-Order Probabilistic Models (FOPM) (4)



- **C-atoms** (später einfach Atom genannt) eines parfactor: Menge instanzierter Zufallsvariablen (z.B.  $p(X, Y)$ ) unter Bedingung (constraint z.B.  $X \neq a$ )
- $RV(\alpha)$  ((ground) random variable  $\rightarrow$  **alle ground atoms**):
  - $\rightarrow g^\theta$  bezeichnet die Anwendung der Substitutionen auf  $g$
  - $\rightarrow$  Menge der Bedingungen  $C_g$  werden auch mit  $\Theta_g$  bezeichnet



Ein FOPM wird durch  $RV(G)$  beschrieben, mit  $G$  als Menge von parfactors

$$P(RV(G)) \propto \prod_{g \in G} \prod_{\theta \in \Theta_g} g^\theta = \Phi(G)$$

Gemeinsame Wahrscheinlichkeit ist Produkt aller Instanzierungen aller parfactors



# Inference Problem

- Gegeben:
  - Menge von Zufallsvariablen  $\rightarrow$  ground atoms, welche für Abfrage (Query)  $Q$  stehen
  - Model  $G$
  - Berechne Wahrscheinlichkeit für  $Q$  gegeben  $G$

- D.h.:

$$P(Q) \propto \sum_{RV(G) \setminus Q} \Phi(G) \left[ = \sum_{RV(G) \setminus Q} \left( \prod_{g \in G} \prod_{\theta \in \Theta_g} g \theta \right) \right]$$

- $RV(G) \setminus Q$  ist die Summe über alle Zuweisungen zu Zufallsvariablen, die nicht in Abfrage  $Q$  vorkommen
- D.h. es gilt die **Atome** zu **eliminieren**  $\rightarrow$  FOVE macht genau das!

# First-Order Variable Elimination Algorithm (FOVE) (1)



- Lösen durch **Elimination**:

→ Ziel von FOVE  
→ Wie geht das?



# First-Order Variable Elimination Algorithm (FOVE) (1)

- Lösen durch **Elimination**:

$$P(Q) \propto \sum_{RV(G) \setminus Q} \Phi(G)$$

- Zur Elimination einer Menge Atome in  $E \rightarrow$  Aufteilen der Summe

$$= \sum_{RV(G) \setminus Q \setminus RV(E)} \sum_{RV(E)} \Phi(G_E) \Phi(G_{\bar{E}})$$

$$= \sum_{RV(G) \setminus Q \setminus RV(E)} \Phi(G_{\bar{E}}) \sum_{RV(E)} \Phi(G_E)$$

- Substitution von  $\sum_{RV(E)} \Phi(G_E)$  durch Potenzialfunktion eines parfactors  $\Phi(g')$ 
  - Reduktion des Modells auf  $G' = G_{\bar{E}} \cup \{g'\} \rightarrow$  weniger Zufallsvariablen

$$P(Q) \propto \sum_{RV(G) \setminus Q \setminus RV(E)} \Phi(G_{\bar{E}} \cup \{g'\}) = \sum_{RV(G') \setminus Q} \Phi(G')$$

# First-Order Variable Elimination Algorithm (FOVE) (1)

- Lösen durch **Elimination**:

$$P(Q) \propto \sum_{RV(G) \setminus Q} \Phi(G)$$

- Zur Elimination einer Menge Atome in

$$= \sum_{RV(G) \setminus Q \setminus RV(E)} \sum_{RV(E)} \Phi(G_E)$$

$$= \sum_{RV(G) \setminus Q \setminus RV(E)} \Phi(G_{\bar{E}}) \sum_{RV(E)} \Phi(G_E)$$

- Substitution von  $\sum_{RV(E)} \Phi(G_E)$  durch Potenzialfunktion eines parfactors  $\Phi(g')$   
➤ Reduktion des Modells auf  $G' = G_{\bar{E}} \cup \{g'\}$  → weniger Zufallsvariablen

$$P(Q) \propto \sum_{RV(G) \setminus Q \setminus RV(E)} \Phi(G_{\bar{E}} \cup \{g'\}) = \sum_{RV(G') \setminus Q} \Phi(G')$$

→ Herausforderung:  
parfactor  $g'$  finden  
→ Durch INVERSION  
ELIMINATION  
→ Durch COUNTING  
ELIMINATION

# First-Order Variable Elimination Algorithm (FOVE) (2)



- FOVE allgemein:
  - Problem exponentieller Laufzeit (entsprechend Anzahl Zufallsvariablen)
  - Lösung: Unabhängigkeit nutzen → Variable Elimination (VE)
  - FOVE ist eine Generalisierung von VE
  - **Vorteil von VE:** Elimination eines Atoms mit seinen Bedingungen entfernt alle groundings. Kosten unabhängig von Anzahl der groundings!
- Vorbedingungen:
  - Das Modell muss bzgl. Abfrage Q **shattered** sein
  - Teilmodell  $G$  muss durch parfactor  $g$  ersetzbar sein (**Fusion**)



# First-Order Variable Elimination Algorithm (FOVE) (3)



## ▪ Shattering:

- Der FOVE Algorithmus nimmt an, dass das FOPM zerrüttet („shattered“) bzgl. Anfrage  $Q$  ist
- Es muss für jedes Paar  $(p, q)$  in  $G$  gelten:
  - Groundings  $RV(p)$  und  $RV(q)$  sind identisch oder disjunkt
  - $(p, q)$  muss in jedem parfactor identisch sein, oder darf niemals instanziiert werden
- Beispiel:
  - Beliebtheit einer Veranstaltung über Zusagen vorhersagen
    - Zusagen, Absagen und keine Antworten → Parfactors dafür definiert
    - FOVE zerlegt („shatters“) nun  $n$  Personen in Gruppen  $n_+$ ,  $n_-$  und  $n_0$





# First-Order Variable Elimination Algorithm (FOVE) (3)



## ▪ Shattering:

- Der FOVE Algorithmus nimmt an, dass das FOPM zerrüttet („shattered“) bzgl. Anfrage  $Q$  ist
- Es muss für jedes Paar  $(p, q)$  in  $G$  gelten:
  - Groundings  $RV(p)$  und  $RV(q)$  sind identisch oder disjunkt
  - $(p, q)$  muss in jedem parfactor identisch sein, oder darf niemals instanziiert werden
- Beispiel:
  - Beliebtheit einer Veranstaltung über Zusagen vorhersagen
    - Zusagen, Absagen und keine Antworten → Parfactors dafür definiert
    - FOVE zerlegt („shatters“) nun  $n$  Personen in Gruppen  $n_+$ ,  $n_-$  und  $n_0$

sonst lässt  
sich Summe  
nicht  
aufteilen...



# First-Order Variable Elimination Algorithm (FOVE) (4)

- **Fusion:** Ersetzung von  $G$  durch parfactor  $g$ . D.h.  $fs(G) = (\phi', A_G, C_G)$

$$\begin{aligned}\Phi(G) &= \prod_{g \in G} \prod_{\theta \in \Theta_g} \phi_g(A_g \theta) = \prod_{g \in G} \prod_{\theta \in \Theta_G} \phi_g(A_g \theta)^{|\Theta_g|/|\Theta_G|} \\ &= \prod_{\theta \in \Theta_G} \prod_{g \in G} \phi_g(A_g \theta)^{|\Theta_g|/|\Theta_G|} = \prod_{\theta \in \Theta_G} \phi'(A_G \theta) = \Phi(fs(G))\end{aligned}$$

- **Menge von parfactors  $G$  wird durch einzelnen ersetzt ( $fs(G)$ )**
- Ersetzung von Lösungen unter Bedingungen  $\Theta_g$  durch Lösungen unter allgemeiner Bedingung  $\Theta_G$ .
  - Dies ermöglicht Schreibweise mit einem Produkt, da gleiche Bedingungen genutzt werden


# First-Order Variable Elimination Algorithm (FOVE) (5)

## ▪ Inversion Elimination:

- Summe aus Produkten  $\rightarrow$  Produkt aus Summen (daher der Name)
- Menge  $\{e\}$  mit  $LV(e) = LV(g)$


$$\theta_1, \dots, \theta_n = \Theta_g$$

$$\begin{aligned} \sum_{RV(e)} \Phi(g) &= \sum_{RV(e)} \prod_{\theta \in \Theta_g} \phi_g(A_g \theta) \\ &= \sum_{e\theta_1} \dots \sum_{e\theta_n} \phi_g(A_g \theta_1) \dots \phi_g(A_g \theta_n) \\ &= \sum_{e\theta_1} \phi_g(A_g \theta_1) \dots \sum_{e\theta_n} \phi_g(A_g \theta_n) \end{aligned}$$

 Shattering

$$= \left( \sum_{e\theta_1} \phi_g(A_g \theta_1) \right) \dots \left( \sum_{e\theta_n} \phi_g(A_g \theta_n) \right)$$

$$\begin{aligned} &= \prod_{\theta \in \Theta_g} \sum_{e\theta} \phi_g(A_g \theta) = \prod_{\theta \in \Theta_g} \sum_{e\theta} \phi_g(A' \theta, e\theta) \\ &= \prod_{\theta \in \Theta_g} \sum_e \phi_g(A' \theta, e) = \prod_{\theta \in \Theta_g} \phi'(A' \theta) \end{aligned}$$

 Fusion

$$= \Phi(g')$$

# First-Order Variable Elimination Algorithm (FOVE) (6)

## ▪ Counting Elimination:

- Anwendung, wenn kein Atom  $e$  in  $G$ , das die Bedingungen erfüllt
  - Auf jeder Menge  $E$  möglich, welche nicht aus ground-atoms besteht
  - $|E| > 1$ , da sonst Inversion anwendbar
  - $A'$  verbleibende ground-atoms
- Auch hier geht es darum einen parfactor zu finden...

$$\sum_{RV(E)} \Phi(g) = \sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi_g(E\theta, A')$$

- $g' = (\phi', A', T)$  ist ein solcher parfactor
- Er definiert eine Potenzialfunktion  $\phi'$  für  $A'$ .

# First-Order Variable Elimination Algorithm (FOVE) (7)

## ▪ Counting Elimination:

➤ Beispiel: mit binären Werten

$$\begin{aligned} & \sum_{RV(p(X))} \prod_{X,Y} \phi(p(X), p(Y)) \\ &= \sum_{RV(p(X))} \phi(0,0)^{|(0,0)|} \phi(1,0)^{|(1,0)|} \phi(1,0)^{|(1,0)|} \phi(1,1)^{|(1,1)|} \end{aligned}$$

- Potenzen sind die Anzahl möglicher Werte für X und Y
- Kombinatorisch können Größen dieser Gruppen berechnet werden

# First-Order Variable Elimination Algorithm (FOVE) (8)



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## ▪ **Der Algorithmus** ...durch sukzessive Eliminationen

- Atom wählen
- Test, ob Inversion möglich (effizientester Fall)
- Wenn nein:
  - Atome zu  $E$  hinzufügen, bis  $G_E$  parfactor, in dem alle Atome mit logischen Variablen Teil von  $E$  sind (Forderung von Counting Elimination)
  - Möglichst viele Inversions finden bevor Counting Elimination ausgeführt wird



# First-Order Variable Elimination Algorithm (FOVE) (9)

- **FOVE**(G,Q)
  - G = Menge von parfactors
  - Q = Teilmenge von RV(G)
  - Q shattered gegenüber Q (Vorbedingung)
  - (1) Wenn  $RV(G) = Q$ , Return (G)
  - (2) **WÄHLE\_ELIMINATION**(G, Q)  $\rightarrow$  (E,  $\{L_1, \dots, L_k\}$ )
  - (3) Fusion: parfactors  $g_E = fs(G_E)$
  - (4) **ELIMINIERE**( $g_E$ , E)  $\rightarrow$  es entsteht  $G'$
  - (5) Return(FOVE( $G'$ , Q))
- **WÄHLE\_ELIMINATION**(G, Q)
  - Wähle e aus  $A_G \setminus Q$
  - $g = fs(G_e)$
  - Wenn  $LV(e) = LV(g)$  und  $\{e\}$  unäre Menge
    - Return( $\{e\}$ , LV(e))
  - Sonst  $E \leftarrow \{e\}$ 
    - Solange  $E \neq$  non-ground atoms von  $G_E$ 
      - $E = E \cup$  non-ground atoms von  $G_E$
  - Return(E, fs( $G_E$ ))
- **ELIMINIERE**( $g_E$ , E)
  - Wenn Inversion möglich, dann führe Inversion durch
  - Sonst: Counting Elimination

## Algorithmus

//Modell = Abfrage  $\rightarrow$  damit schon beantwortet

// $G' = G_E \cup \{g\}$

//Fusion

# Fazit



- Modellbeschreibungen oft first-order
- Meist werden sie dann auf propositionale Ebene herunter gebrochen und mit einem propositionalen Algorithmus gelöst
- FOVE ist eine Generalisierung von VE
- FOVE eliminiert nacheinander Zufallsvariablen
- FOVE behält die first-order Struktur bei, was die Berechnung beschleunigt
- FOVE kann auf mehr Fälle angewendet werden als Algorithmus von Poole





# Ausblick (1)

- Counting Elimination ist teurer als Inversion, sollte aber auch seltener vorkommen
  - Es existieren aber vermutlich bessere Methoden, welche es zu finden gilt...
- Es existieren bereits Erweiterungen:
  - C-FOVE verbessert Laufzeit durch Einsatz von Counting Formulas
  - Sie fassen Abhängigkeiten zwischen großen Mengen an Variablen kompakt zusammen (anwendbar, wenn es nur auf Anzahl der Instanzen ankommt → z.B. ist eine Veranstaltung überfüllt?)
- Weitere Forschung mit folgender Zielrichtung:
  - Approximative Inference
  - Abfragen mit Nicht-ground-atoms

# Ausblick (2)

- Approximative Inference:
  - Benutzer kann zwischen Genauigkeit und Effizienz wählen (→ trade off)
  - Grundlage ist probabilistische Datenbank (keine first-order Repräsentation)
  - Verwendung von speziellen Graphen
    - I. Ähnlichkeiten werden erkannt und für lifting genutzt → Partitionierung des Graphen
      - Pfadlängen im Graph können als Parameter vom Anwender angepasst werden
    - II. Distanzmaße werden genutzt, um Ähnlichkeiten festzustellen
      - Distanzmaße sind frei definierbar
    - III. Variation von VE
      - Nicht über alle Faktoren multiplizieren, sondern über Teilmengen

- L. Getoor und B. Taskaar. **Introduction to Statistical Relational Learning.** *MIT Press, 2007*
- R. de Salvo Braz, E. Amir und D. Roth. **Lifted First-Order Probabilistic Inference.** *Department of Computer Science, University of Illinois, 2007*
- R. de Salvo Braz, E. Amir und D. Roth. **MPE and Partial Inversion in Lifted Probabilistic Variable Elimination.** *Department of Computer Science, University of Illinois, 2006*
- D. Poole. **First-Order Probabilistic Inference.** *Department of Computer Science, University of British Columbia, 2003*
- K. Kersting, B. Milch, L. S. Zettlemoyer, M. Haimes und L. P. Kaelbling. **Reasoning about Large Populations with Lifted Probabilistic Inference.** *MIT CSAIL, Cambridge, 2007*
- K. Kersting, B. Milch, L. S. Zettlemoyer, M. Haimes und L. P. Kaelbling. **Lifted Probabilistic Inference with Counting Formulas.** *MIT CSAIL, Cambridge, 2008*
- G. F. Cooper. **The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks.** *Knowledge Systems Laboratory, Stanford, 1990*
- P. Sen, A. Deshpande, L. Getoor. **Bisimulation-based Approximative Lifted Inference.** *Computer Science Department, University of Maryland*

---

# Danke für Eure Aufmerksamkeit

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



# First-Order Variable Elimination Algorithm (FOVE) (Anhang 1)

## ▪ Counting Elimination:

### ▪ Gegeben:

- Unabhängige Atome (z.B.  $p(X_1), p(X_2)$ ) unter einer Bedingung C  
→ d.h. Wahl einer Substitution hat keine Auswirkung auf andere mögliche Substitutionen

### ▪ Multinomialer Zähler:

- a sei Atom → multinomialer Zähler  $\vec{N}_a$  ist ein Vektor, dessen Werte  $N_{a,j}$  angeben, wie viele Instanzen von a den j-ten Wert in  $D_a$  haben
- Multinomialkoeffizient ist eine Verallgemeinerung des Binomialkoeffizienten und gibt Anzahl der Zuweisungen in  $RV(a)$  an

$$\vec{N}_a! = \frac{(\vec{N}_{a,1} + \dots + \vec{N}_{a,|D_a|})!}{\vec{N}_{a,1}! \dots \vec{N}_{a,|D_a|}!}$$

$\vec{N}_A$  ist Zähler für Menge A von Atomen

$$\prod_{a \in A} \vec{N}_a! = \vec{N}_A!$$

# First-Order Variable Elimination Algorithm (FOVE) (Anhang 2)



## ▪ Counting Elimination-Theorem:

- **Counting elimination** bringt Verbesserungen der Performanz, da das Iterieren über Zuweisungen exponentiell ist in  $RV(E)$ , während es über Gruppen von Zuweisungen polynomial ist.

- $g$  sei shattered parfactor
- $E = \{E_1, \dots, E_k\} \subset A_g$
- $RV(E)$  disjunkt zu  $RV(A_g \setminus E) \rightarrow$  wir bezeichnen  $A_g \setminus E$  mit  $A'$
- $A'$  sind alle ground atoms, in denen jedes Paar unabhängig ist gegeben Bedingung  $C_g$
- Dann gilt:

$$\sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi(A_g \theta) = \sum_{\vec{N}_E} \vec{N}_E! \prod_{v \in D_E} \phi(v, A')^{\prod_{i=1}^k \vec{N}_{E_i, v_i}}$$



# First-Order Variable Elimination Algorithm (FOVE) (Anhang 3)



## Algorithmus

<p>FUNCTION <i>FOVE</i>(<math>G, Q</math>)  <math>G</math> a set of parfactors, <math>Q \subseteq RV(G)</math>, <math>G</math> shattered against <math>Q</math></p> <ol style="list-style-type: none"> <li>1. If <math>RV(G) = Q</math>, return <math>G</math>.</li> <li>2. <math>(E, \{L_1, \dots, L_k\}) \leftarrow \text{CHOOSE-ELIMINATION}(G, Q)</math>.</li> <li>3. <math>g_E \leftarrow fs(G_E)</math></li> <li>4. <math>G' \leftarrow \text{ELIMINATE}(g_E, E)</math>.</li> <li>5. Return <math>\text{FOVE}(G' \cup G_{-E}, Q)</math>.</li> </ol>
<p>FUNCTION <i>ELIMINATE</i>(<math>g, E, \{L_1, \dots, L_k\}</math>)</p> <ol style="list-style-type: none"> <li>1. If <math>k = 0</math> (no inversion)  return <math>\text{SUMMATION-WITHOUT-INVERSION}(g, E)</math>.</li> <li>2. <math>E_1 \leftarrow \{e \in E : LV(e) \cap L_1 \neq \emptyset\}</math> (get inverted atoms).</li> <li>3. Return <math>\bigcup_{C_1 \in U_L(C_g)} \text{ELIMINATE-GIVEN-UNIFORMITY}(g, E_1, C_1, \{L_2, \dots, L_k\})</math>.</li> </ol>
<p>FUNCTION <i>ELIMINATE-GIVEN-UNIFORMITY</i>(<math>g, E_1, C_1, \{L_2, \dots, L_k\}</math>)</p> <ol style="list-style-type: none"> <li>1. Choose <math>\theta_1 \in [C_1]</math> (bind inverted logical variables arbitrarily).</li> <li>2. <math>G' \leftarrow \text{ELIMINATE}(g\theta_1, E_1\theta_1, \{L_2, \dots, L_k\})</math>.</li> <li>3. <math>G'' \leftarrow \bigcup_{g'\theta_1 \in G'} (\phi_{g'}, A_{g'}, C_1 \wedge C_{g'})</math>.</li> <li>4. Return <math>\bigcup_{g'' \in G''} \text{SIMPLIFICATION}(g'')</math></li> </ol>
<p>FUNCTION <i>SUMMATION-WITHOUT-INVERSION</i>(<math>g, E</math>)</p> <ol style="list-style-type: none"> <li>1. If <math>E = \{E_1, \dots, E_k\}</math> atoms are independent given <math>C_g</math> and <math>A_g \setminus E</math> is ground  return <math>(\sum_{\vec{N}_E} \vec{N}_E! \prod_v \phi_g(v, A_g \setminus E) \prod_{i=1}^k \vec{N}_{E_i, v_i}, A_g \setminus E, \top)</math></li> <li>2. Return <math>(\sum_{RV(E)} \prod_{\theta \in \Theta_g} \phi_g(A_g\theta_g), A_g \setminus E, C_g)</math> (propositional elimination).</li> </ol>
<p>Notation:</p> <ul style="list-style-type: none"> <li>▪ <math>LV(\alpha)</math>: logical variables in object <math>\alpha</math>.</li> <li>▪ <math>g\theta</math>: parfactor <math>(\phi_g, A_g\theta, C_g\theta)</math>.</li> <li>▪ <math>U_L(C_g)</math>: USCP of <math>L</math> with respect to <math>C_g</math></li> <li>▪ <math>C_{ L}</math>: constraints projected to a set of logical variables <math>L</math>.</li> <li>▪ <math>G_E</math>: subset of parfactors <math>G</math> which depend on <math>RV(E)</math>.</li> <li>▪ <math>G_{-E}</math>: subset of parfactors <math>G</math> which do not depend on <math>RV(E)</math>.</li> <li>▪ <math>\top</math>: tautology constraint.</li> </ul>

<p>FUNCTION <i>CHOOSE-ELIMINATION</i>(<math>G, Q</math>)</p> <ol style="list-style-type: none"> <li>1. Choose <math>e</math> from <math>A_G \setminus Q</math>.</li> <li>2. <math>g \leftarrow fs(G_e)</math></li> <li>3. If <math>LV(e) = LV(g)</math> and <math>\forall e' \in A_g RV(e') \neq RV(e)</math>  return <math>(\{e\}, LV(e))</math> (inversion eliminable).</li> <li>4. <math>E \leftarrow \{e\}</math>.</li> <li>5. While <math>E \neq</math> non-ground atoms of <math>G_E</math>  <math>E \leftarrow E \cup</math> non-ground atoms of <math>G_E</math>.</li> <li>6. Return <math>(E, \text{GET-SEQUENCE-OF-INVERSIONS}(fs(G_E)))</math>.</li> </ol>
<p>FUNCTION <i>GET-SEQUENCE-OF-INVERSIONS</i>(<math>g</math>)</p> <ol style="list-style-type: none"> <li>1. If there is no <math>L_1</math> set of invertible logical variables in <math>g</math>  return <math>\emptyset</math>.</li> <li>2. Choose <math>\theta_1 \in [C_{g L_1}]</math>.</li> <li>3. <math>\{L_2, \dots, L_k\} \leftarrow \text{GET-SEQUENCE-OF-INVERSIONS}(g\theta_1)</math>.</li> <li>4. Return <math>\{L_1, L_2, \dots, L_k\}</math>.</li> </ol>

One possible way of choosing an elimination.

