Lifted Junction Tree Algorithm

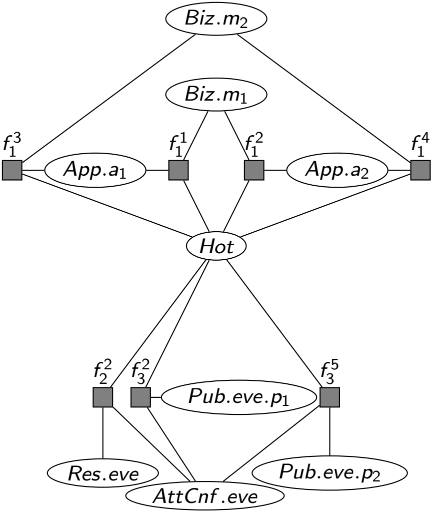
Counting and Conjunctive Queries

Tanya Braun, Ralf Möller

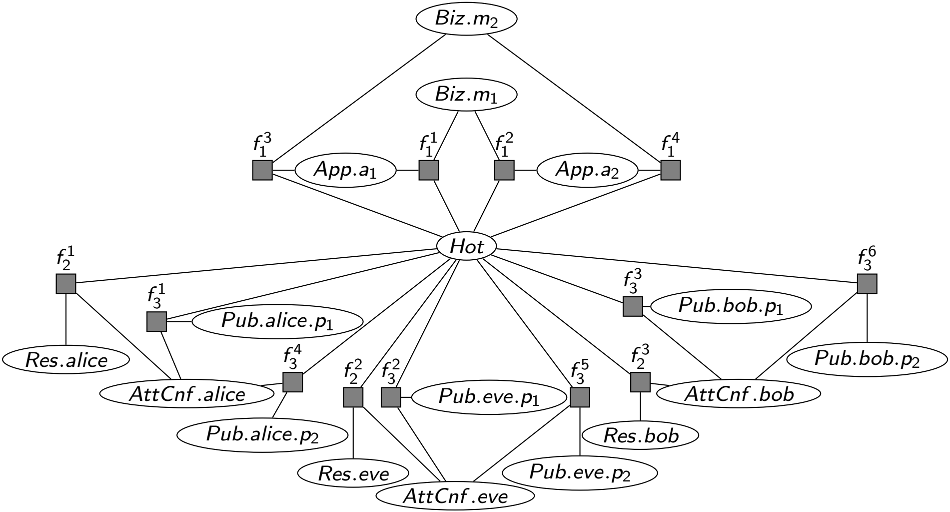
Institute of Information Systems
University of Lübeck

August 21, 2017

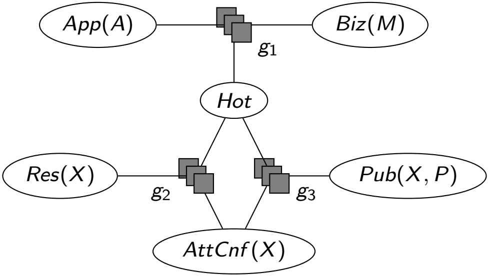
Problem: Propositional Models Explode



Problem: Propositional Models Explode



Approach: Parameterization and Lifted Inference for Query Answering (QA)



Propositional
$\rightarrow$ 17 random variables
$\rightarrow$ 13 factors

Lifted
$\rightarrow$ 6 (parameterized) random variables
$\rightarrow$ 4 domains $\mathcal{D}$ of logical variables
$\rightarrow$ 3 parametric factors

Next Problem: Make Query Answering More Practical

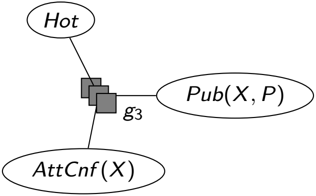
Workshop Contribution

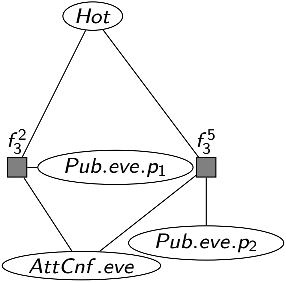
Extensions to
Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

- Fewer groundings by exploiting technique of counting
- From multiple atomic queries to multiple conjunctive queries

Lifted Variable Elimination (LVE) – Single Queries

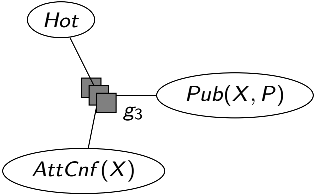
Poole (2003), de Salvo Braz (2007), Milch et al. (2008), Taghipour (2013)





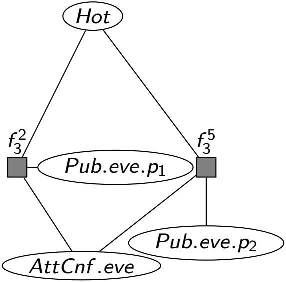
Lifted Variable Elimination (LVE) – Single Queries

Poole (2003), de Salvo Braz (2007), Milch et al. (2008), Taghipour (2013)



$$f_3^2 = \phi(Hot, AttCnf.eve, Pub.eve.p_1)$$

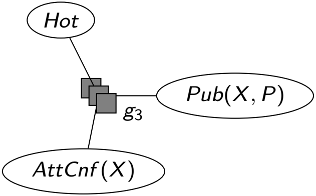
Hot	*AttCnf.eve*	*Pub.eve.p*$_1$	ϕ
true	*true*	*true*	0.4
true	*true*	*false*	0.2
⋮			



Overall, 8 cases

$$\sum_{v \in range(Pub.eve.p_1)} \phi(Hot, AttCnf.eve, v)$$

LVE: Avoiding Grounding by Exponentiation



$$g_3 = \phi(Hot, AttCnf(X), Pub(X, P))$$

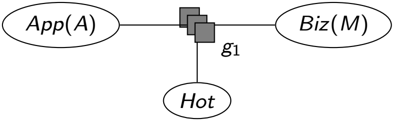
Hot	*AttCnf(X)*	*Pub(X, P)*	ϕ
true	*true*	*true*	0.4
true	*true*	*false*	0.2
⋮			

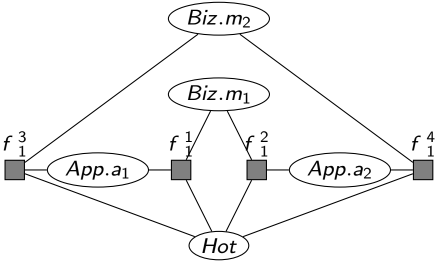
Overall, 8 entries

$$\sum_{Pub(X,P)} g_3 := \left(\sum_{\substack{v \in \\ range(Pub(X,P))}} \phi(Hot, AttCnf(X), v) \right)^{|\mathcal{D}(P)|}$$

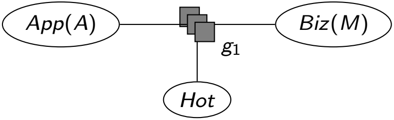
$Pub(X, P)$ contains all logical variables in g_3.

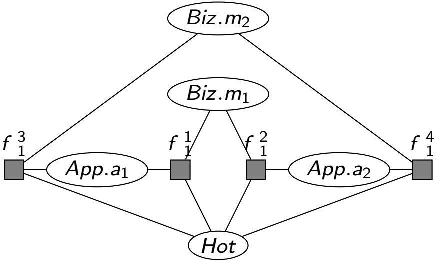
LVE: Avoiding Grounding by Counting





LVE: Avoiding Grounding by Counting





$$\sum_{\substack{v\in \\ range(App.a_1)}} \phi(Hot, v, Biz.m_1, Biz.m_2) \rightarrow \text{16 cases}$$

$$= \sum_{\substack{v\in \\ range(App.a_1)}} \phi(Hot, v, \#_M[Biz(M)])$$

LVE: Count-conversion

$g_1 = \phi(Hot, App(A), Biz(M))$

Hot	$App(A)$	$Biz(M)$	ϕ
$true$	$true$	$true$	0.4
$true$	$true$	$false$	0.2
$\vdots$			

Overall, 8 cases

$g_1' = \phi(Hot, App(A), \#_M[Biz(M)])$

Hot	$App(A)$	$\#$	ϕ
$true$	$true$	$[2, 0]$	0.4^2
$true$	$true$	$[1, 1]$	$0.4 \cdot 0.2$
$true$	$true$	$[0, 2]$	0.2^2
$\vdots$			

Overall, 12 cases (instead of 16)

LVE: Count-conversion

$g_1 = \phi(Hot, App(A), Biz(M))$

Hot	$App(A)$	$Biz(M)$	ϕ
true	*true*	*true*	0.4
true	*true*	*false*	0.2
⋮			

Overall, 8 cases

$g_1' = \phi(Hot, App(A), \#_M[Biz(M)])$

Hot	$App(A)$	$\#$	ϕ
true	*true*	$[2, 0]$	0.4^2
true	*true*	$[1, 1]$	$0.4 \cdot 0.2$
true	*true*	$[0, 2]$	0.2^2
⋮			

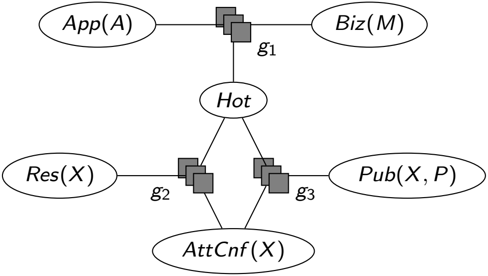
Overall, 12 cases (instead of 16)

$$\sum_{App(A)} g_1 := \left(\sum_{\substack{v \in \\ range(App(A))}} \phi(Hot, v, \#_M[Biz(M)]) \right)^{|\mathcal{D}(A)|}$$

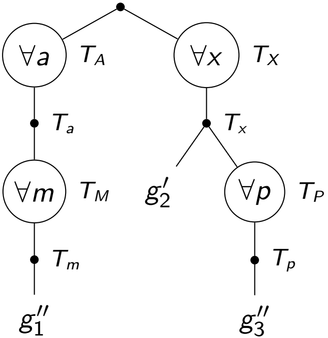
First-order Decomposition Trees (FO Dtrees)

Darwiche (2001), Taghipour (2013)

Model

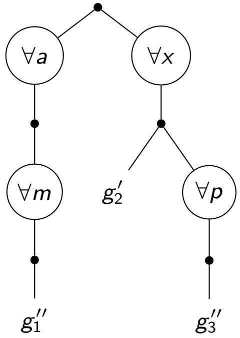


FO dtree

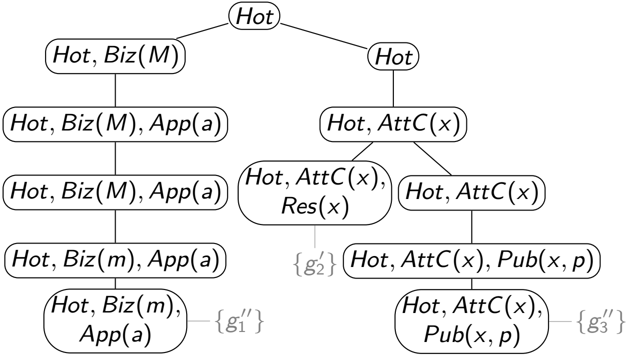


First-order Junction Trees (FO Jtrees)

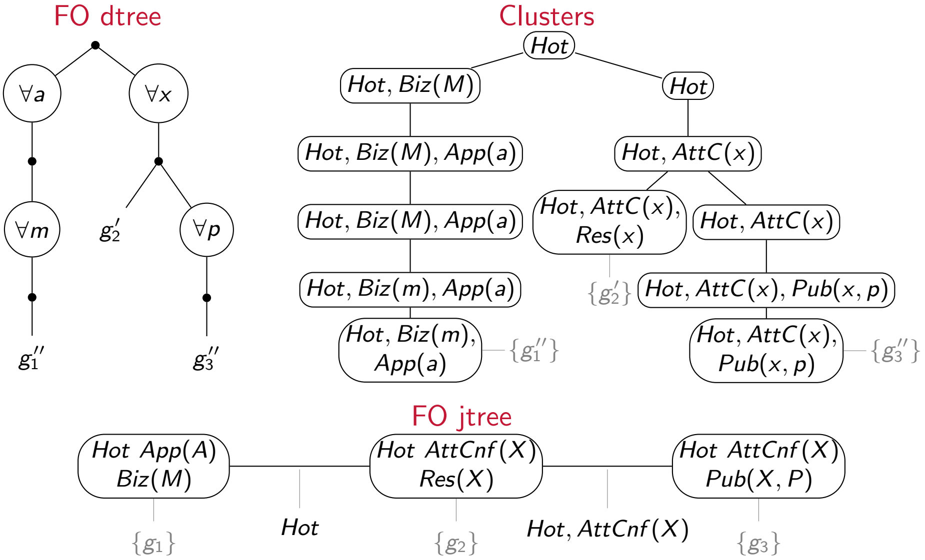
FO dtree



Clusters

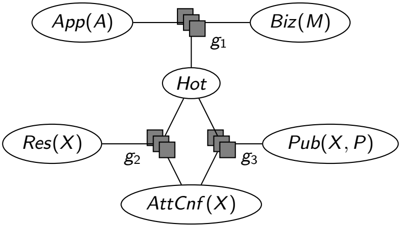


First-order Junction Trees (FO Jtrees)

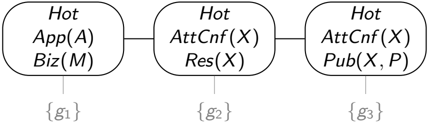


LJT – Multiple Queries

Lauritzen & Spiegelhalter (1988), Koller & Friedman (2009), Braun & Möller (2016)



Atomic queries on ground random variables, e.g., $Res.eve$, $Pub.eve.p_1$



Input

- Model G
- Queries $\mathbf{Q}$

Algorithm

1. Build FO jtree for G.
2. Pass messages
   - inbound
   - outbound
3. Answer queries $\mathbf{Q}$.

(without evidence)

Extended LJT: Example

Input + Construction

Model $G = \{g_1, g_2, g_3\}$,

- $g_1 = \phi_1(Hot, App(A), Biz(M))$
- $g_2 = \phi_2(Hot, AttCnf(X), Res(X)))$
- $g_3 = \phi_3(Hot, AttCnf(X), Pub(X, P))$

Queries $\mathbf{Q} = \{Q_1, Q_2\}$

- $Q_1 = P(Pub(eve, p_1))$ (atomic query)
- $Q_2 = P(Pub(eve, p_1), Res(eve))$ (conjunctive query)

Extended LJT: Example

Input + Construction

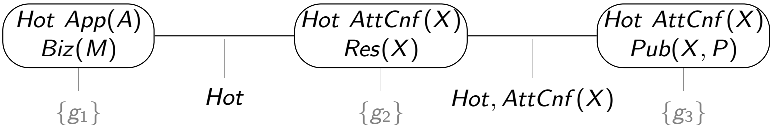
Model $G = \{g_1, g_2, g_3\}$,

- $g_1 = \phi_1(Hot, App(A), Biz(M))$
- $g_2 = \phi_2(Hot, AttCnf(X), Res(X)))$
- $g_3 = \phi_3(Hot, AttCnf(X), Pub(X, P))$

Queries $\mathbf{Q} = \{Q_1, Q_2\}$

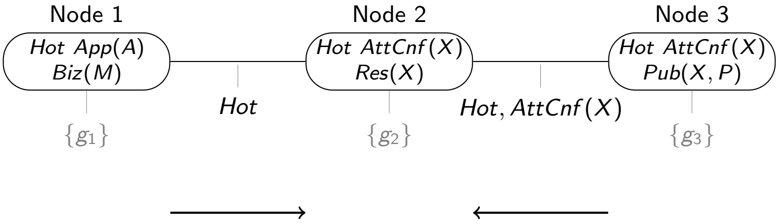
- $Q_1 = P(Pub(eve, p_1))$ (atomic query)
- $Q_2 = P(Pub(eve, p_1), Res(eve))$ (conjunctive query)

Constructed FO jtree



Extended LJT: Example Continued

Message Passing

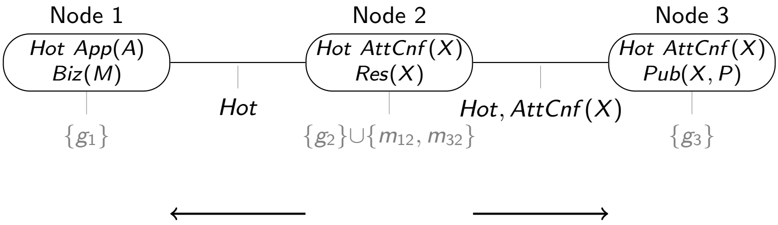


$$m_{12} = \sum_{\#_M[Biz(M)]} \sum_{App(A)} g_1$$

$$m_{32} = \sum_{Pub(X,P)} g_3$$

Extended LJT: Example Continued

Message Passing

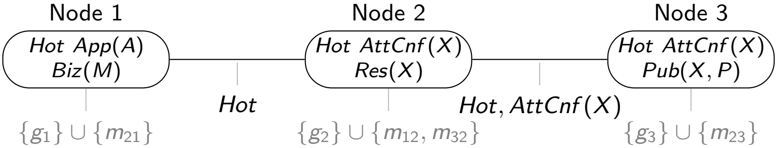


$$m_{21} = \sum_{AttCnf(X)} m_{32} \sum_{Res(X)} g_2$$

$$m_{23} = m_{12} \sum_{Res(X)} g_2$$

Extended LJT: Example Continued

Query Answering



Query

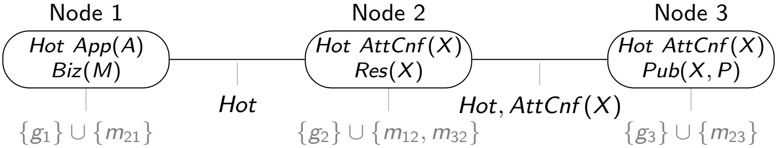
$Q = P(Pub(eve, p_1))$

Submodel

$G^Q = \{g_3\} \cup \{m_{23}\}$

Extended LJT: Example Continued

Query Answering



Query

$Q = P(Pub(eve, p_1))$

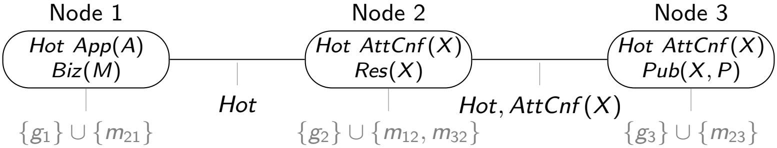
Submodel

$G^Q = \{g_3\} \cup \{m_{23}\}$

$$Q \propto \sum_{Hot} \sum_{AttCnf(X)} m_{23} \sum_{\substack{Pub(X,P),\\ X \neq eve,\\ P \neq p_1}} g_3$$

Extended LJT: Example Continued

Query Answering



Query

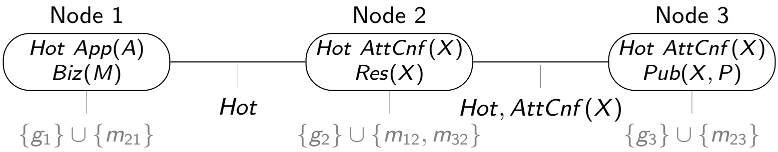
$Q = P(Pub(eve, p_1), Res(eve))$

Submodel

$G^Q = \{g_2\} \cup \{g_3\} \cup \{m_{12}\}$

Extended LJT: Example Continued

Query Answering



Query

$$Q = P(Pub(eve, p_1), Res(eve))$$

Submodel

$$G^Q = \{g_2\} \cup \{g_3\} \cup \{m_{12}\}$$

$$Q \propto \sum_{Hot} m_{12} \sum_{AttCnf(X)} \sum_{\substack{Res(X),\\ X \neq eve}} g_2 \sum_{\substack{Pub(X,P),\\ X \neq eve,\\ P \neq p_1}} g_3$$

Analysis: LVE vs. LJT

LVE

- Eliminates complete model for each query
- Influence on QA: number of PRVs in model

LJT

- Finds subtree, eliminates smaller model in subtree
- Influence on QA: number of PRVs in submodel

Analysis: LVE vs. LJT

LVE

- Eliminates complete model for each query
- Influence on QA: number of PRVs in model

LJT

- Finds subtree, eliminates smaller model in subtree
- Influence on QA: number of PRVs in submodel

LJT preprocessing necessary $\rightarrow$ Static overhead

- New construction if structure changes
- New message passing if domain sizes or values change

PRV = parameterized random variable

Test Run: LVE vs. LJT

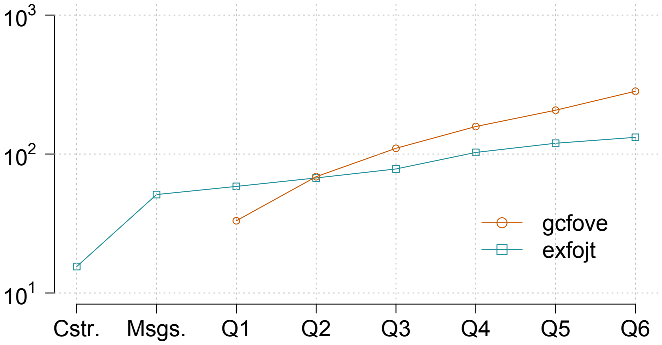


Figure: Accumulated Runtimes [ms] over 6 queries (points connected for readability)

`gcfove`: Implementation of LVE by Taghipour (2013)
`exfojt`: Own implementation of extended LJT

Problem: Make Query Answering More Practical

Workshop Contribution

Extensions to
Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

- Fewer groundings by exploiting technique of counting
- From multiple atomic queries to multiple conjunctive queries

Problem: Make Query Answering More Practical

Workshop Contribution

Extensions to
Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

- Fewer groundings by exploiting technique of counting
- From multiple atomic queries to multiple conjunctive queries

Future Directions

- Incrementally changing models
- Dynamic variant

Cutset, Context, Cluster

Darwiche (2001), Taghipour (2013)

$$cutset(T) = \bigcup_{T_i, T_j \in child(T)} RV(T_i) \cap RV(T_j) \setminus acutset(T)$$

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

$$context(T) = RV(T) \cap acutset(T)$$

$$cluster(T) = cutset(T) \cup context(T)$$

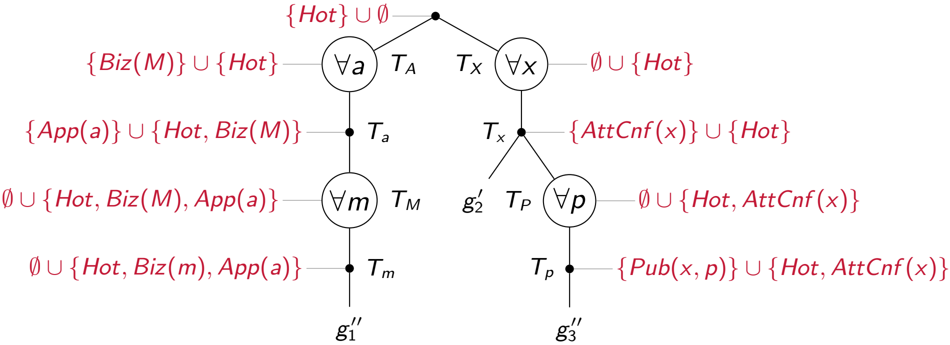
$$cluster(L) = RV(\phi_L), L \text{ leaf}$$

$$RV(T) = \bigcup_{T' \in child(T)} RV(T')$$

$$RV(L) = RV(\phi_L), L \text{ leaf}$$

Cutset, Context, Cluster

Darwiche (2001), Taghipour (2013)



Labels: $cutset(N) \cup context(N)$

Cutset, Context, Cluster

Darwiche (2001), Taghipour (2013), Labels: $cutset(N) \cup context(N)$

$$cutset(T) = \bigcup_{T_i, T_j \in child(T)} RV(T_i) \cap RV(T_j) \backslash acutset(T), acutset(T) = \bigcup_{T' \in ancestor(T)} cutset(T')$$

$$context(T) = RV(T) \cap acutset(T), RV(T) = \bigcup_{T' \in child(T)} RV(T'), RV(L) = RV(\phi_L), L \text{ leaf}$$

$$cluster(T) = cutset(T) \cup context(T), cluster(L) = RV(\phi_L), L \text{ leaf}$$