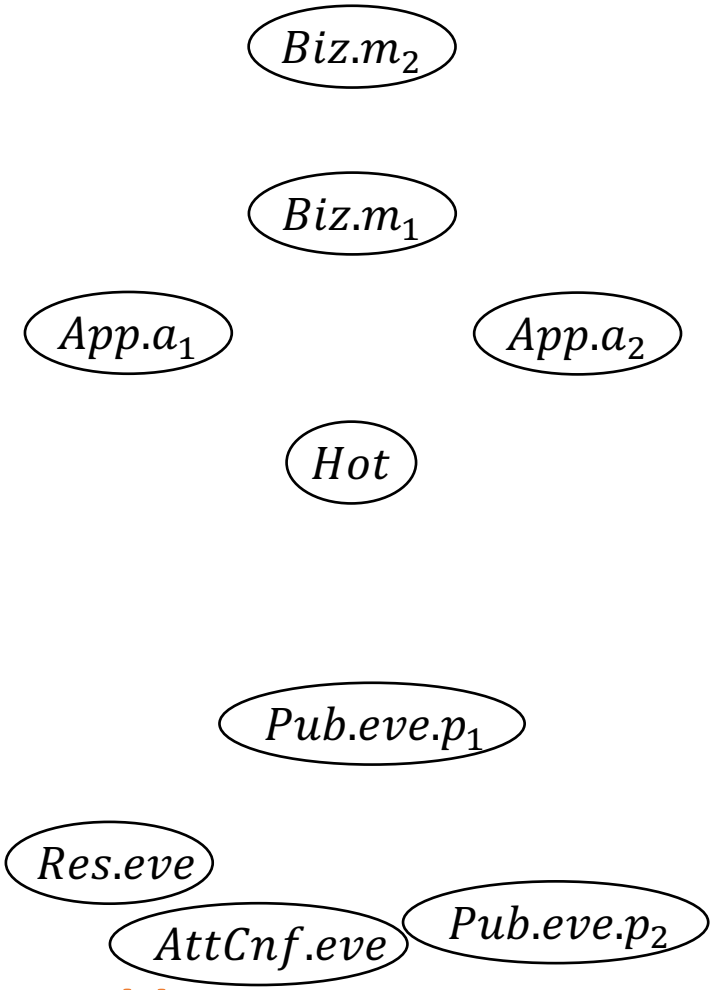
# Lifted Junction Tree Algorithm

Tanya Braun

Institute for Information Systems

# Propositional Models: Worlds

- Characterise world by random variables
  - E.g., $Hot$
  - Possible values (range): $r(Hot) = \{0,1\}$
- 1 world = specific values for variables
- Potential per world
- Joint probability distribution $P_G$ over all possible worlds

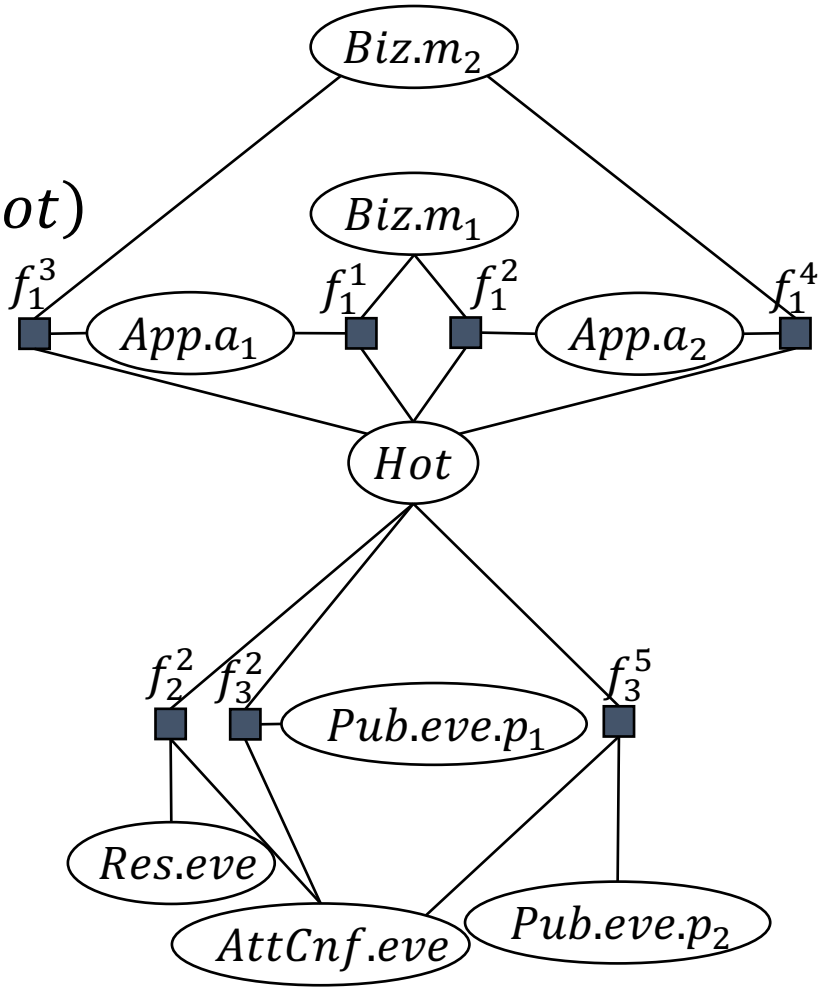



**$2^9 = 512$ possible worlds**

# Propositional Models: Factors

- Full joint $P_G$ as product of factors
  - $f_2^2(Res.eve, AttCnf.eve, Hot)$

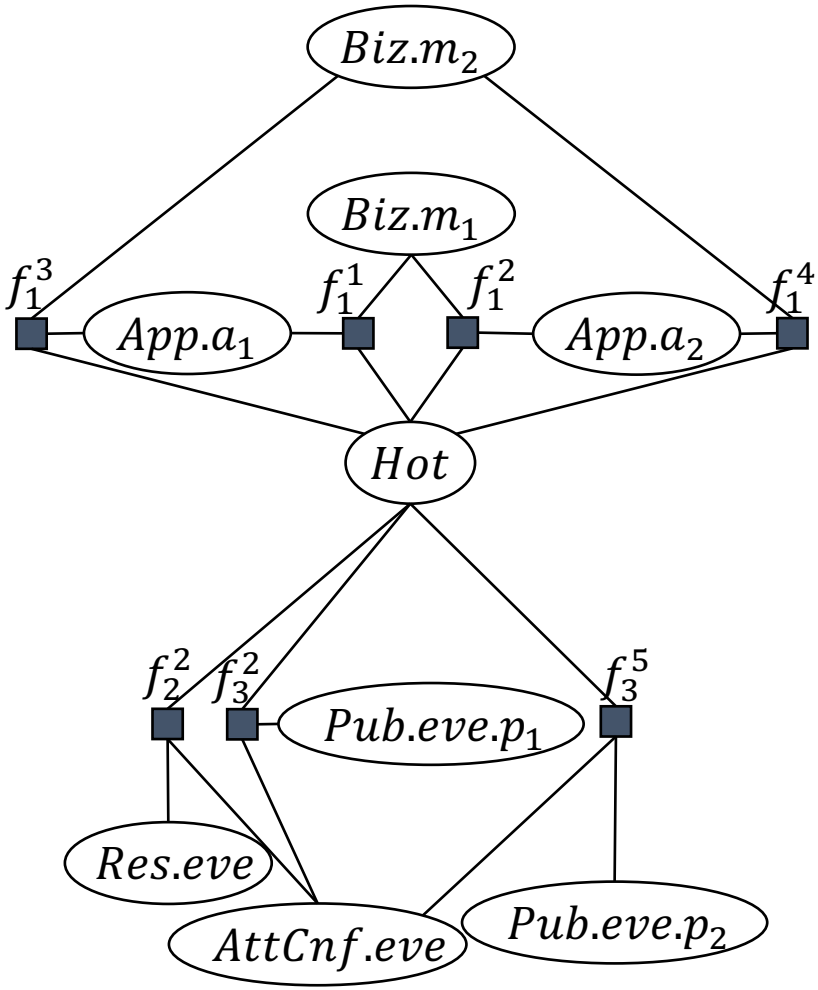
| $Res.eve$ | $Hot$ | $AttCnf.eve$ | $f_2^2$ |
|---|---|---|---|
| 0 | 0 | 0 | 5 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 6 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 6 |
| 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 9 |




$7 \cdot 2^3 = 56$ entries

# Query Answering (QA): Queries

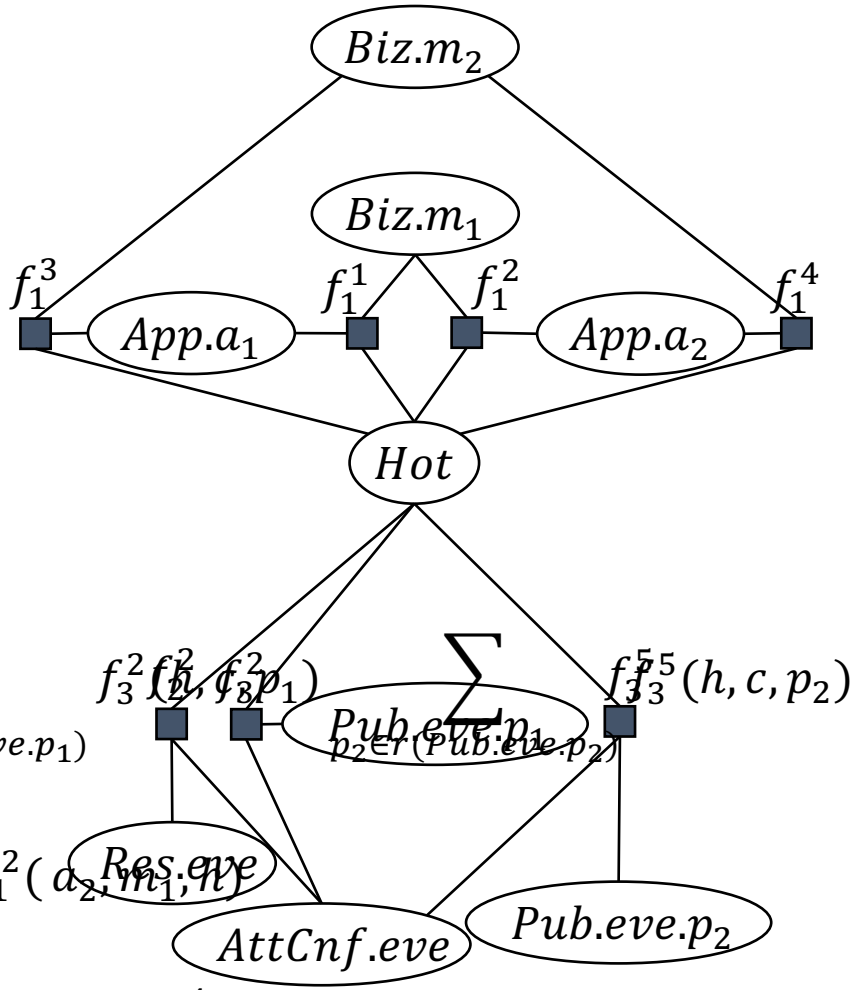
- Probability of an event
  - $P(Res.eve = 1)$
- Marginal distribution
  - $P(Res.eve)$
  - $P(Res.eve, Pub.eve.p_1)$
- Conditional distribution
  - $P(Res.eve|Hot = 1)$
- Most probable assignment
  - $\underset{rv(G)}{\text{argmax}}\, P_G$
  - $\underset{Res.eve,Biz.m_1}{\text{argmax}}\, P_G$



# QA: Variable Elimination (VE)

- Eliminate all variables not appearing in query
  - Through summing out
- E.g., marginal
  - $P(Res.eve)$
  - Sum out all non-query variables

Zhang and Poole (1994)




$$\sum_{h \in r(Hot)} \sum_{c \in r(AttCnf.eve)} f_2^2(Res.eve, h, c) \sum_{p_1 \in r(Pub.eve.p_1)} f_3^2(h, c, p_1) \sum_{p_2 \in r(Pub.eve.p_2)} f_3^5(h, c, p_2)$$

$$\sum_{a_1 \in r(App.a_1)} \sum_{a_2 \in r(App.a_2)} \sum_{b_1 \in r(Biz.m_1)} f_1^1(a_1, m_1, h) f_1^2(a_2, m_1, h)$$

$$\sum_{b_2 \in r(Biz.m_2)} f_1^3(a_1, m_2, h) f_1^4(a_2, m_2, h)$$

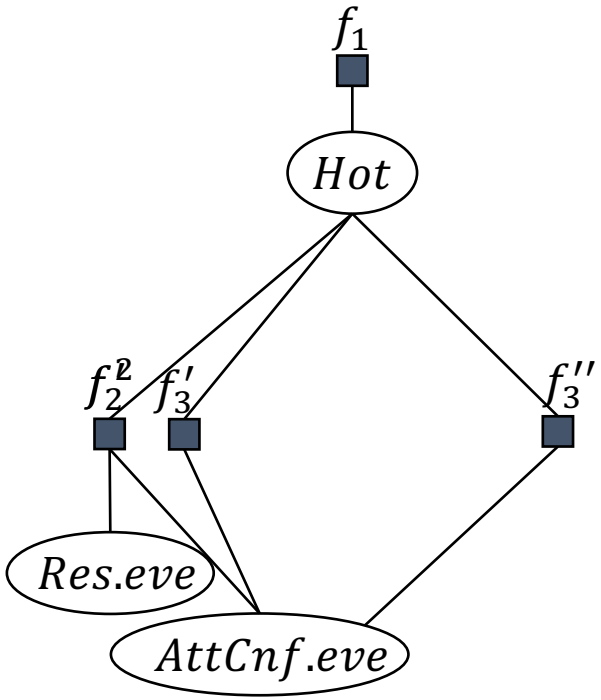
# QA: Variable Elimination (VE)

Zhang and Poole (1994)

$$P(Res.eve) \propto \sum_{h \in r(Hot)} f_1(h) \sum_{c \in r(AttCnf.eve)} f_2^2(Res.eve, c, h) f_3'(h, c) f_3''(h, c)$$

$$= \sum_{h \in r(Hot)} f_1(h) \sum_{c \in r(AttCnf.eve)} f'(Res.eve, h, c)$$

| $Res.eve$ | $Hot$ | $AttCnf.eve$ | $f'$ |
|---|---|---|---|
| 0 | 0 | 0 | 50 |
| 0 | 0 | 1 | 10 |
| 0 | 1 | 0 | 34 |
| 0 | 1 | 1 | 16 |
| 1 | 0 | 0 | 54 |
| 1 | 0 | 1 | 36 |
| 1 | 1 | 0 | 12 |
| 1 | 1 | 1 | 69 |

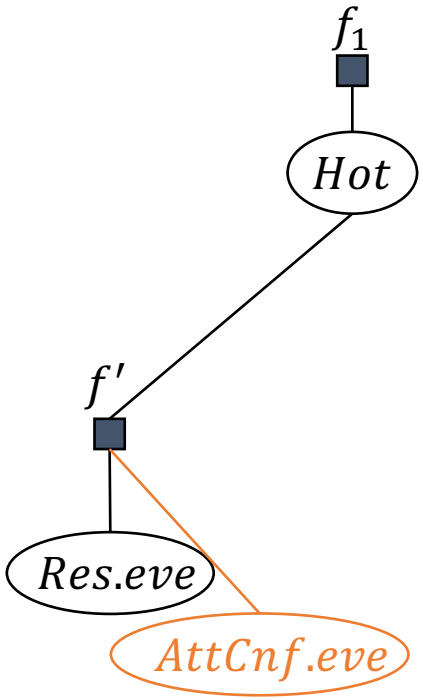


# QA: Variable Elimination (VE)

$$P(Res.eve) \propto \sum_{h \in r(Hot)} f_1(h) \sum_{c \in r(AttCnf.eve)} f_2^2(Res.eve, c, h) f_3'(h, c) f_3''(h, c)$$

$$= \sum_{h \in r(Hot)} f_1(h) \sum_{c \in r(AttCnf.eve)} f'(Res.eve, h, c)$$

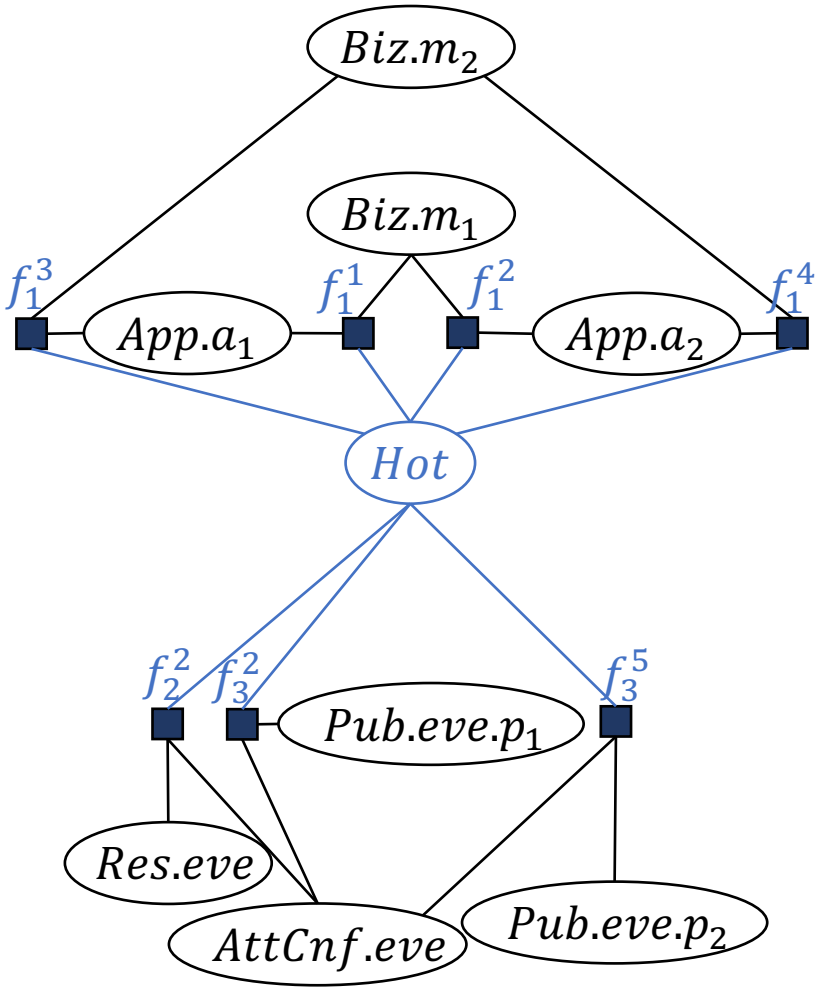
| $Res.eve$ | $Hot$ | ~~$AttCnf.eve$~~ | $f'$ | $\Sigma$ |
|---|---|---|---|---|
| 0 | 0 | ~~0~~ | 50 | 60 |
| 0 | 0 | ~~1~~ | 10 | |
| 0 | 1 | ~~0~~ | 34 | 50 |
| 0 | 1 | ~~1~~ | 16 | |
| 1 | 0 | ~~0~~ | 54 | 90 |
| 1 | 0 | ~~1~~ | 36 | |
| 1 | 1 | ~~0~~ | 12 | 81 |
| 1 | 1 | ~~1~~ | 69 | |



# QA: Variable Elimination (VE)

- Eliminate all variables not appearing in query
  - Through summing out
- E.g., conditional
  - $P(Res.eve|Hot = 1)$
  - Absorb $Hot = 1$ in all factors
  - Sum out remaining variables

Zhang and Poole (1994)

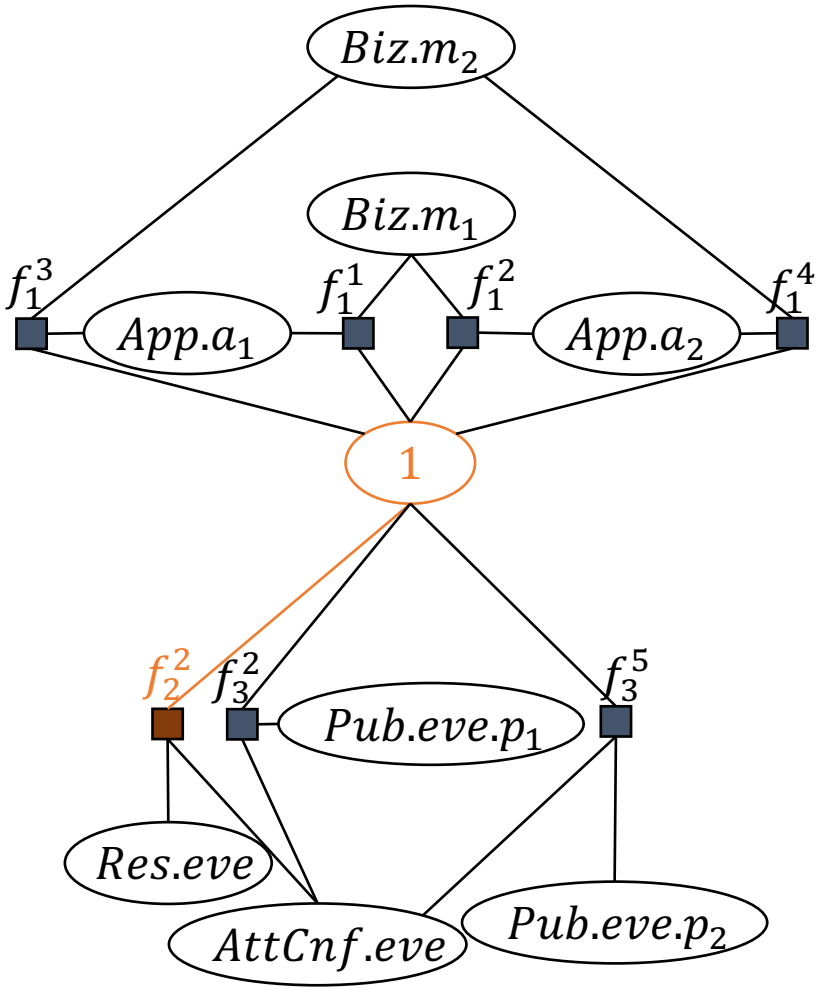


# QA: Evidence Absorption

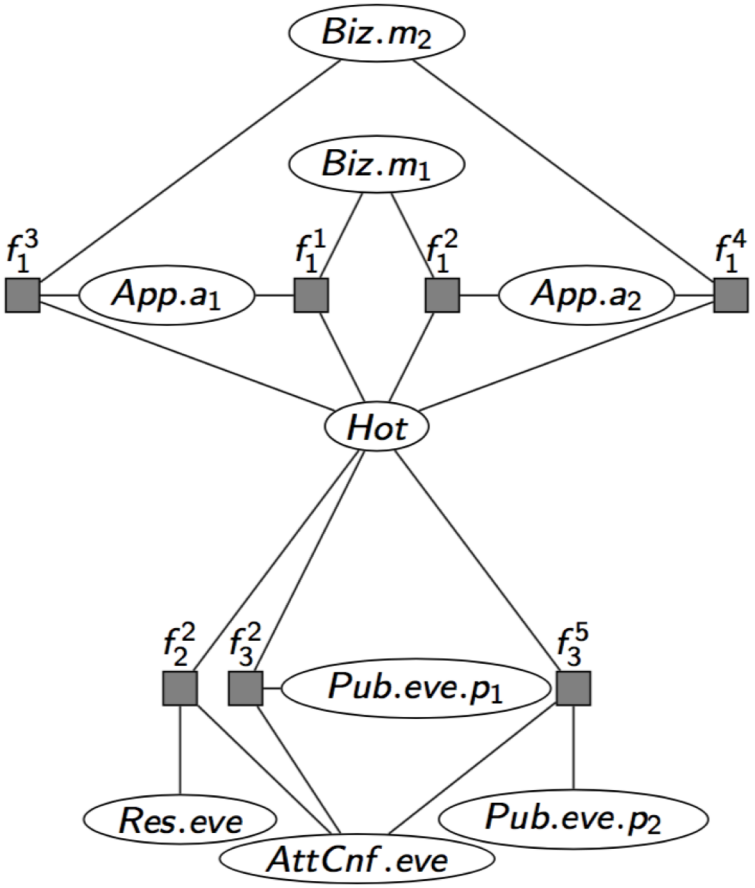
- For conditional queries
  - Absorb $Hot = 1$ in all factors, e.g., $f_2^2$
  - Possiby eliminate variable

| $Res.eve$ | $Hot$ | $AttCnf.eve$ | $f_2^2$ |
|---|---|---|---|
| 0 | 0 | 0 | ~~5~~ 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 6 |
| 1 | 0 | 0 | ~~4~~ 0 |
| 1 | 0 | 1 | ~~6~~ 0 |
| 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 9 |

Zhang and Poole (1994)

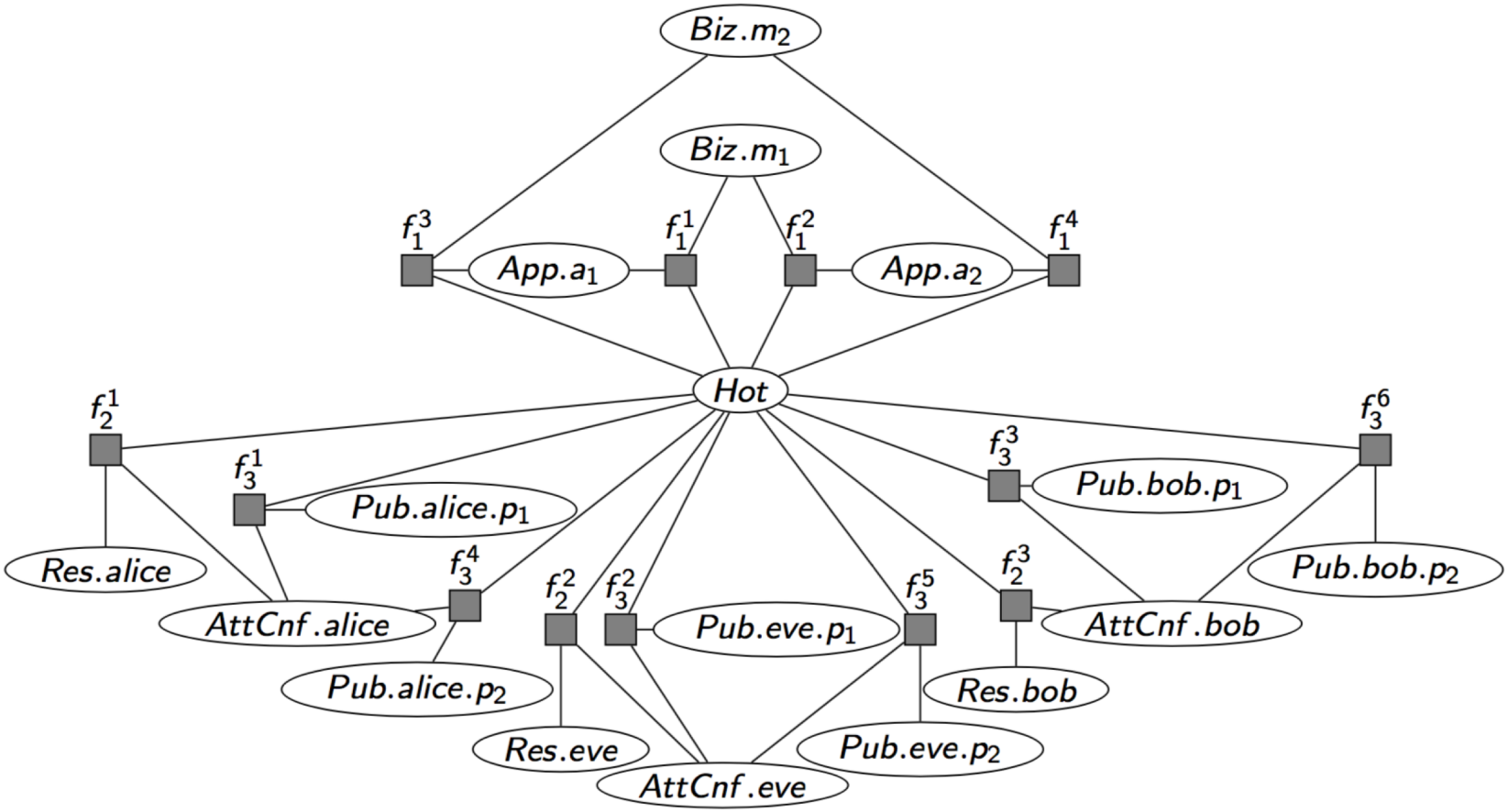


# Problem: Models Explode




**$7 \cdot 2^3 = 56$ entries in 7 factors, 9 variables**

# Problem: Models Explode

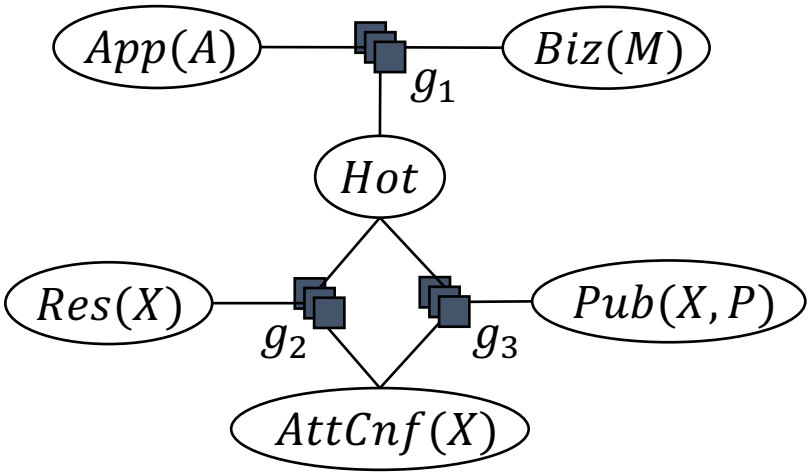



**$13 \cdot 2^3 = 104$ entries in 13 factors, 17 variables**

# Solution: Lifting

Poole and Zhang (2003)

- Parameterised random variables = PRV
  - With logical variables
  - E.g., $X$
  - Possible values (domain): $\mathcal{D}(X) = \{alice, eve, bob\}$

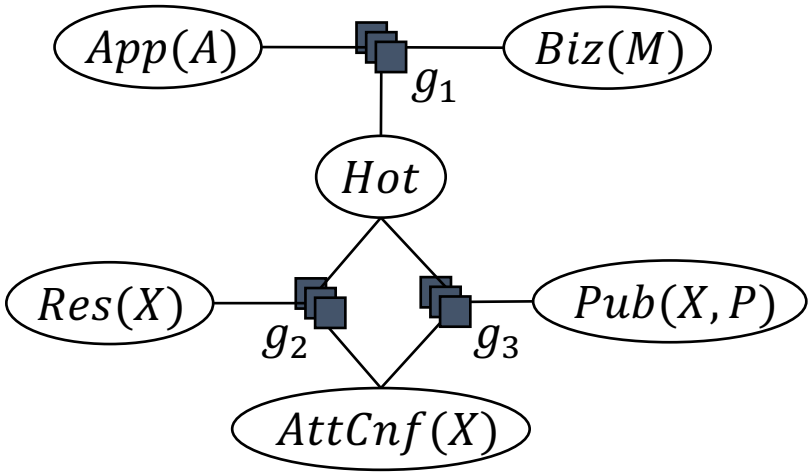


# Solution: Lifting

Poole and Zhang (2003)

- Factors with PRVs = parfactors
  - E.g., $g_2$

| $Res(X)$ | $Hot$ | $AttCnf(X)$ | $g_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 5 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 6 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 6 |
| 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 9 |

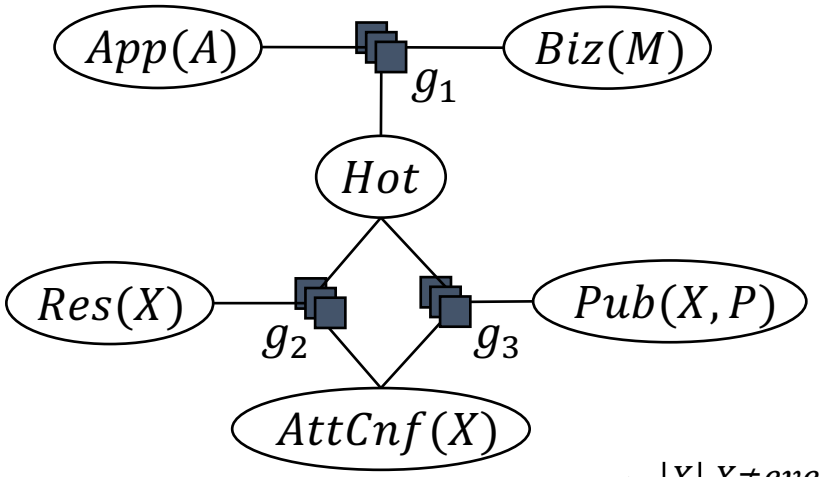



$3 \cdot 2^3 = 24$ entries in 3 factors, 6 PRVs

# QA: Lifted VE (LVE)

Taghipour et al. (2013)

- Eliminate all variables not appearing in query
  - Through lifted summing out
  - Exponentiate result for isomorphic instances
- E.g., marginal
  - $P(Res(eve))$
  - Sum out all non-query variables



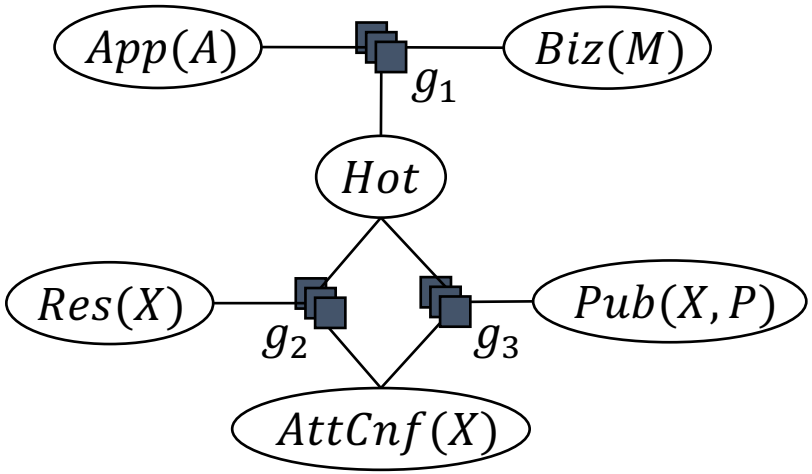

$$\sum_{h \in r(Hot)} \left( \sum_{c \in r(Res(X))} \sum_{c \in r(AttCnf(X))} g_2\,(Res.eve, H, c) \left( \sum_{p \in r(Pub(X,P))} g_3(h,c,p) \right)^{|P|} \right)^{|X|, X \neq eve}$$

$$\left( \sum_{a \in r(App(A))} \sum_{m \in r(Biz(M))} g_1(a,m,h) \right)^{*}$$

* Counting magic

# Problem: Many Queries

- Set of queries
  - $P(Res(eve))$
  - $P(AttCnf(eve))$
  - $P(Pub(eve, p_1))$
  - $P(Hot)$
  - $P(App(a_1))$
  - $P(Biz(m_1))$
  - Combinations of variables
- Under evidence
  - $AttCnf(X') = 1$
  - $X' \in \{alice, eve\}$

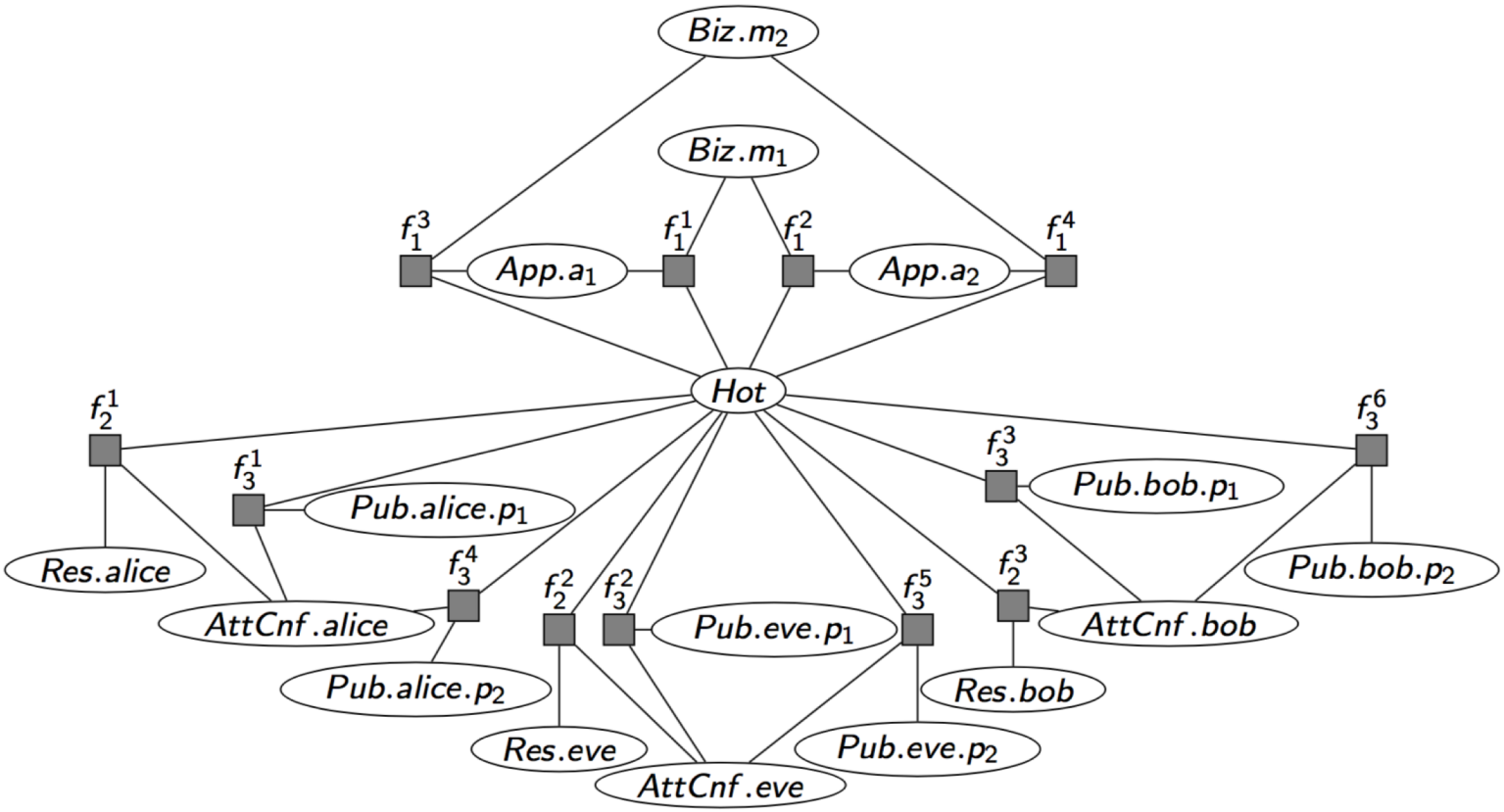



- (L)VE starts with complete model for QA

# Solution: Junction Tree Algorithm

Lauritzen and Spiegelhalter (1988)

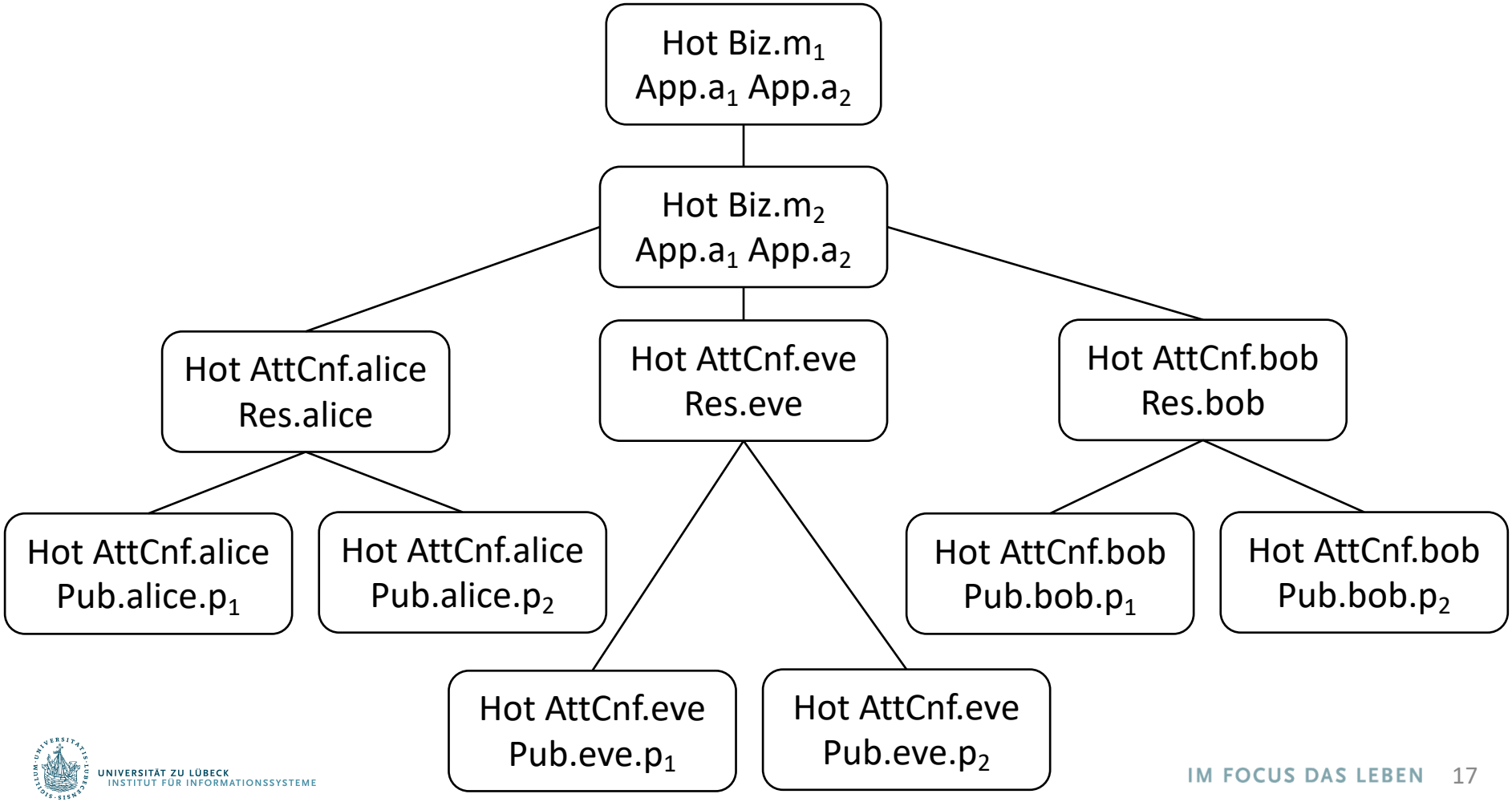
- Identify clusters in model to form junction tree



# Junction Tree: Data Structure

Lauritzen and Spiegelhalter (1988)

- Clusters: sets of variables from underlying model



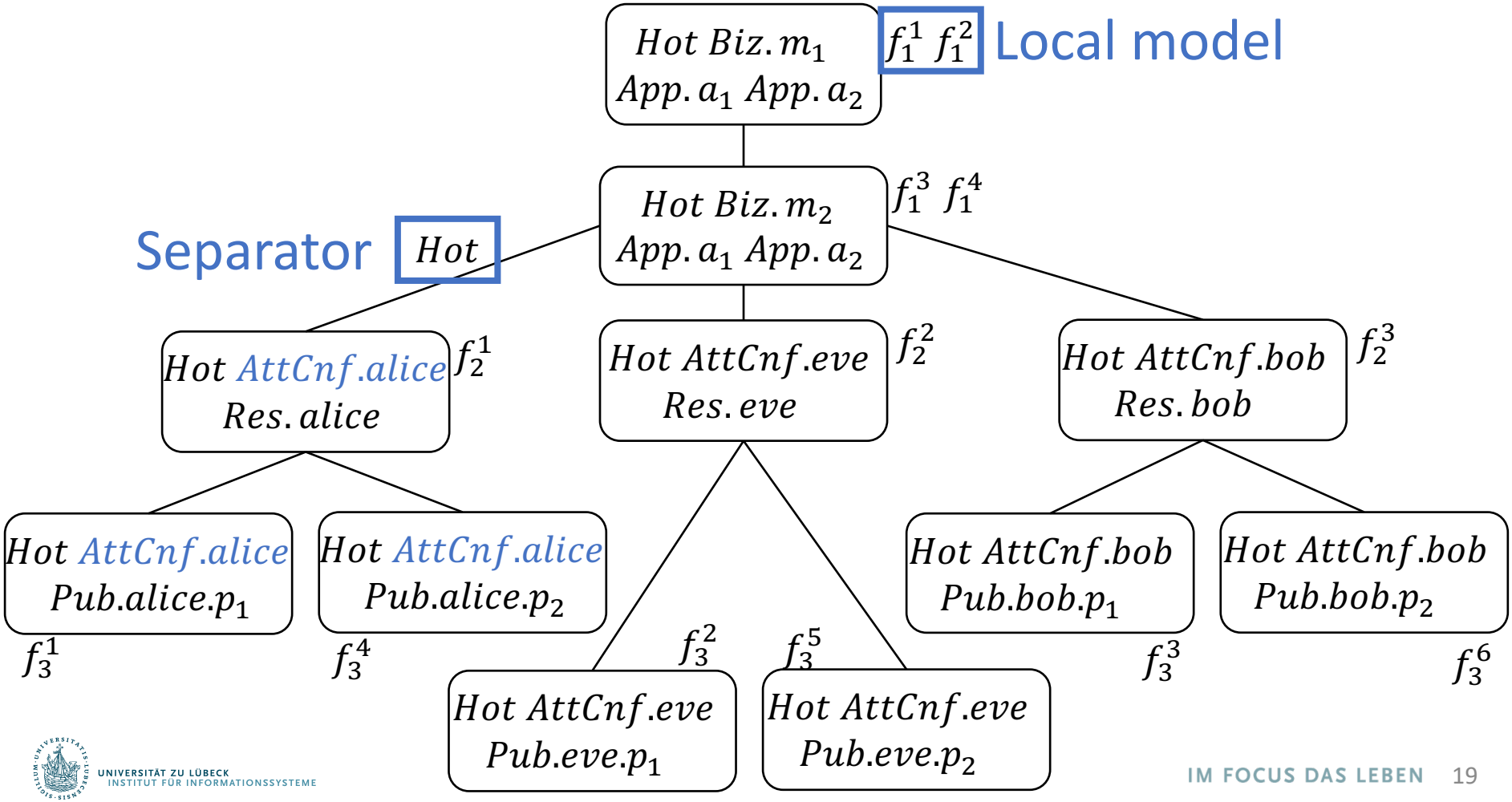
# Junction Tree: Definition

Lauritzen and Spiegelhalter (1988)

- Junction tree $J = (V, E)$ for model $F$
  - $V$ set of nodes, a node = cluster $C$
  - $E$ set of edges
  1. A cluster $C_i$ is a set of variables from $F$
  2. For every factor $f$ in $F$, its arguments appear in some cluster $C_i$
  3. If a variable from F appears in clusters $C_i$ and $C_j$, it must appear in every cluster $C_k$ on the path between nodes $i$ and $j$.
- Each cluster has a local model: set of factors
  - factor arguments subset of cluster
- Separator: shared variables of edges $(i, j) \in E$

# Junction Tree: Components

- Clusters sufficient for QA after some preprocessing



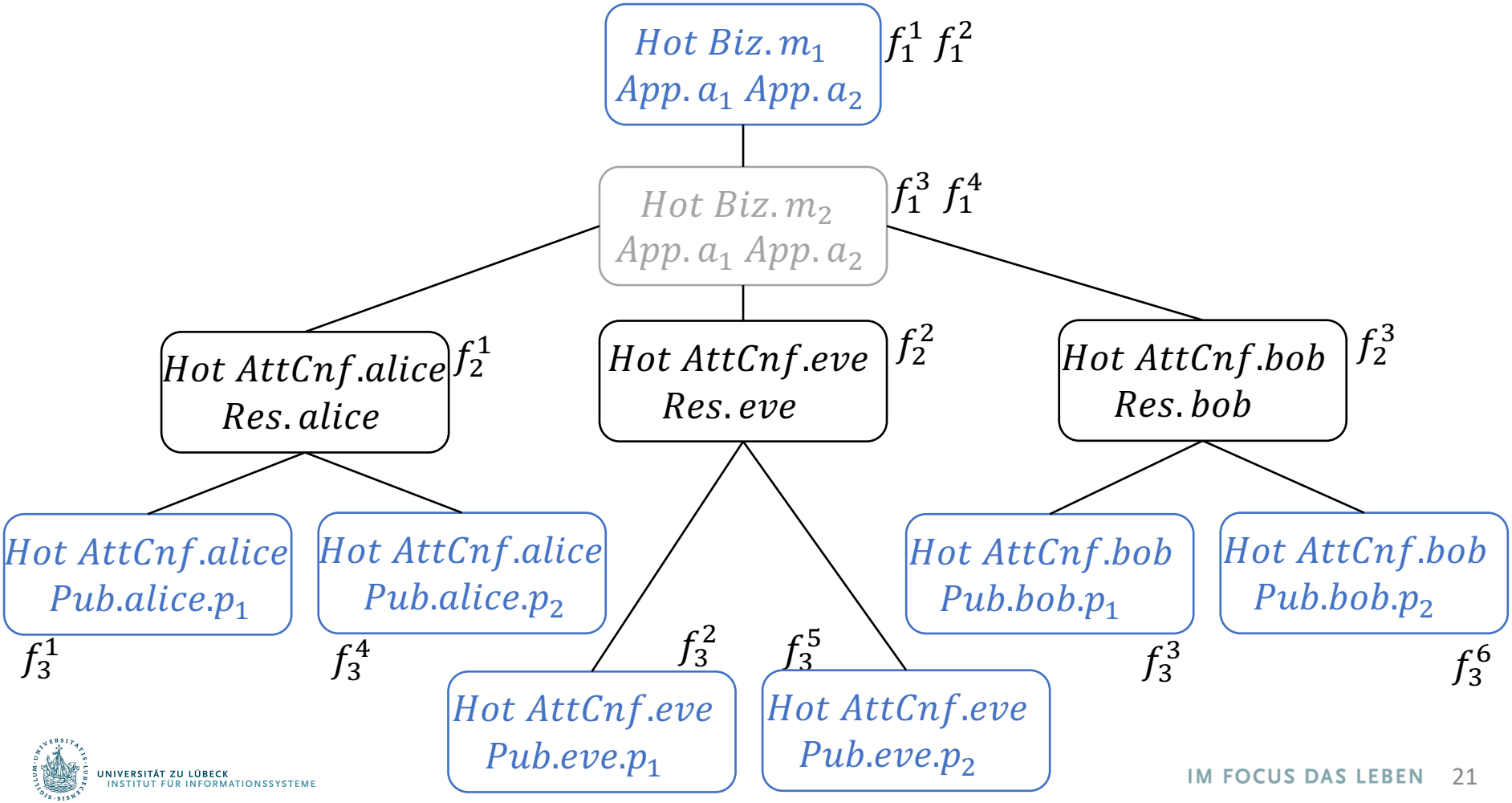
# Junction Tree: Message Passing

Lauritzen and Spiegelhalter (1988), Shenoy and Shafer (1990)

- Distribute local information by messages
- Messages from periphery to centre and back
  - VE on local model and other messages
  - I.e., query over separator variables
- Local computations correctness:
  - Valid junction tree
  - combination & marginalisation
    (in the form of multiplication & summing out)
- QA on local models (including messages)
  - Use cluster that contains query variables

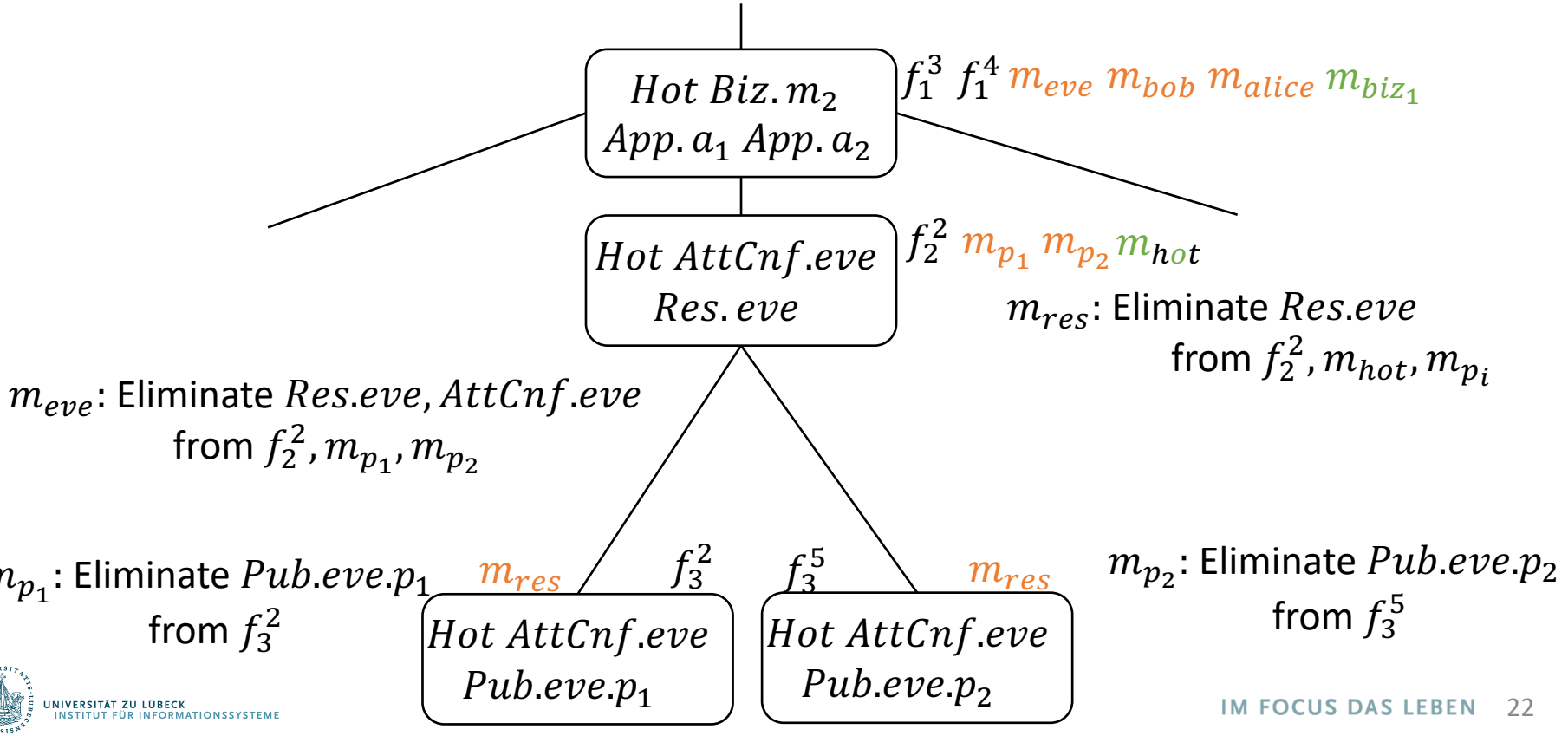
# Junction Tree: Messages

- From periphery to centre and back



# Junction Tree: Symmetries

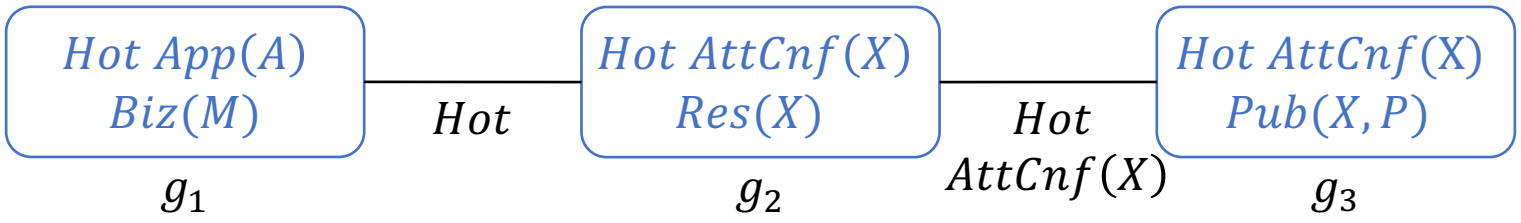
- Identical messages
- Information already present



# First-order Junction Tree: FO jtree

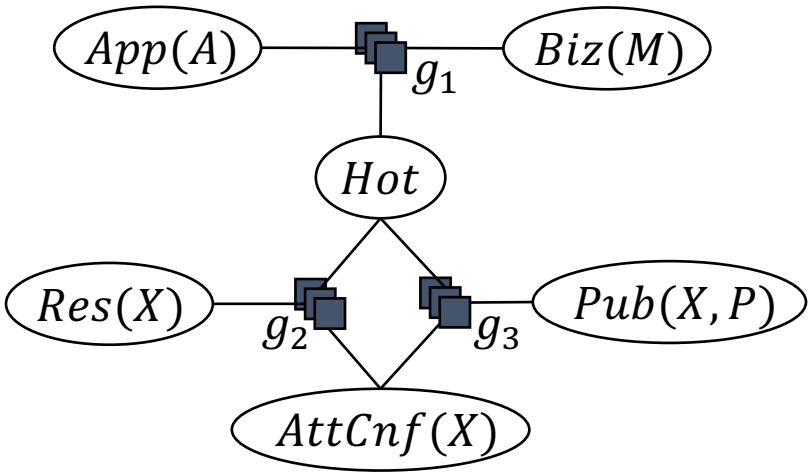
Braun and Möller (2016)

- Parameterised clusters = parclusters
  - Nodes of FO jtree
  - Shared PRVs between neighbours = separators
  - Parclusters have local models of parfactors
- FO jtree for parfactor model $G$:
  - Analogous definition, junction tree properties apply
  - Message passing, QA analogous

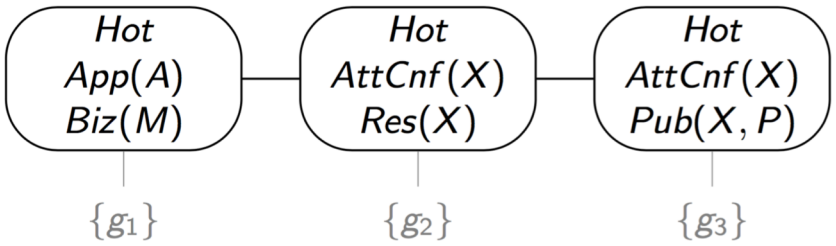


# Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017, 2017a)




Queries on grounded PRVs, e.g.,
$Res(eve), Pub(eve, p_1), Hot$




- Input
  - Model $G$
  - Evidence $\boldsymbol{E}$
  - Queries $\boldsymbol{Q}$
- Algorithm
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $\boldsymbol{E}$ into $J$
  3. Pass messages in $J$
     - Inbound
     - Outbound
  4. Answer queries $\boldsymbol{Q}$

# LJT: Example Input

- Model $G = \{g_i\}_{i=1}^{3}$
  - $g_1\big(Hot, App(A), Biz(M)\big)$
  - $g_2(Res(X), Hot, AttCnf(X))$
  - $g_3\big(Hot, AttCnf(X), Pub(X, P)\big)$

  → Including function specification
- Evidence $\boldsymbol{E} = \{AttCnf(alice) = 1, AttCnf(eve) = 1\}$
- Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$

- Algorithm
  1. Build FO jtree $J$ for $G$

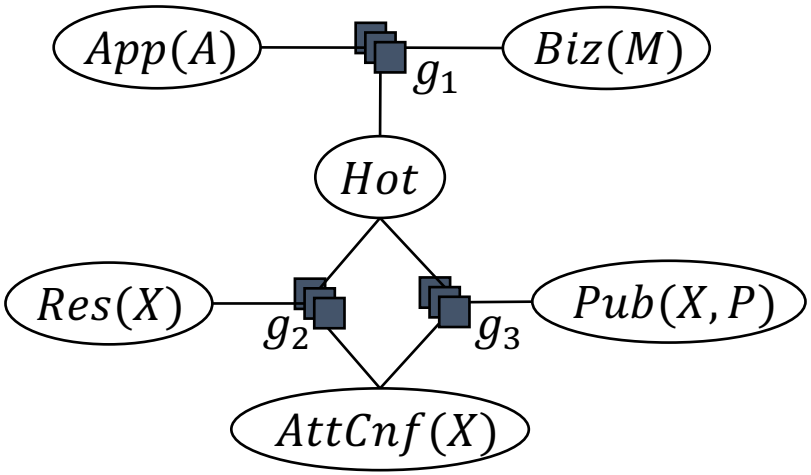
# FO Jtree Construction

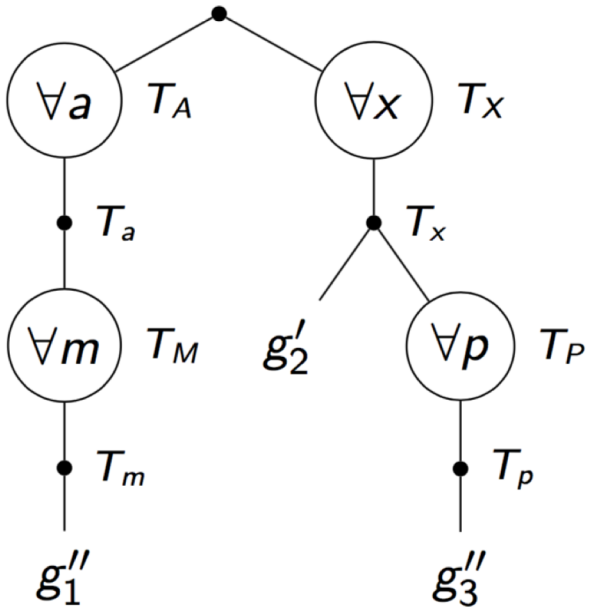
- Propositional jtree construction
  - Triangulation, compute maximum spanning tree, …
  - Hypergraph partitioning
- First-order: logical variables
  - First-order decomposition trees (FO dtrees)
  - FO dtrees have node properties (cutset, context, cluster)
  - (FO) dtree + clusters = (FO) jtree
    *(works in propositional case, too)*
  - Heuristic to build an FO dtree
    *(logical variables guide the construction)*

# FO Dtree: Data Structure

Taghipour et al. (2013)

- (L)VE decomposes model into subproblems
- Represent as tree



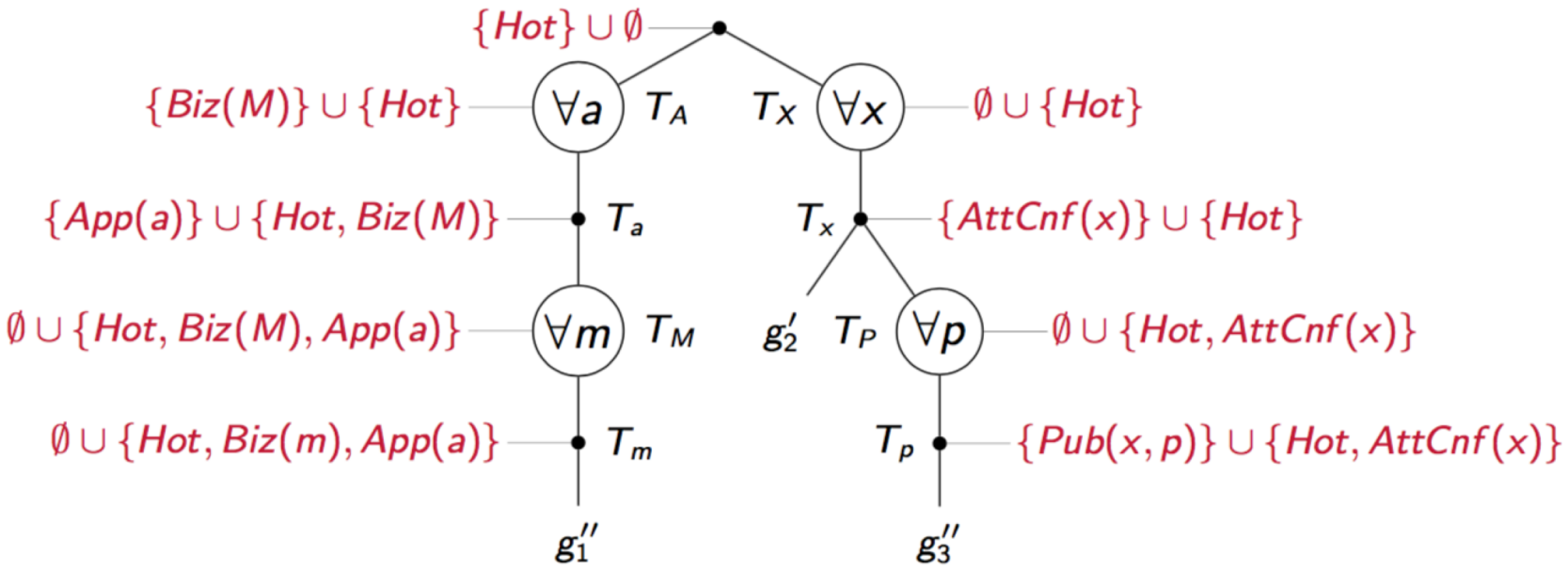
# FO Dtree: Cutset, Context, Cluster

Taghipour et al. (2013a)

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

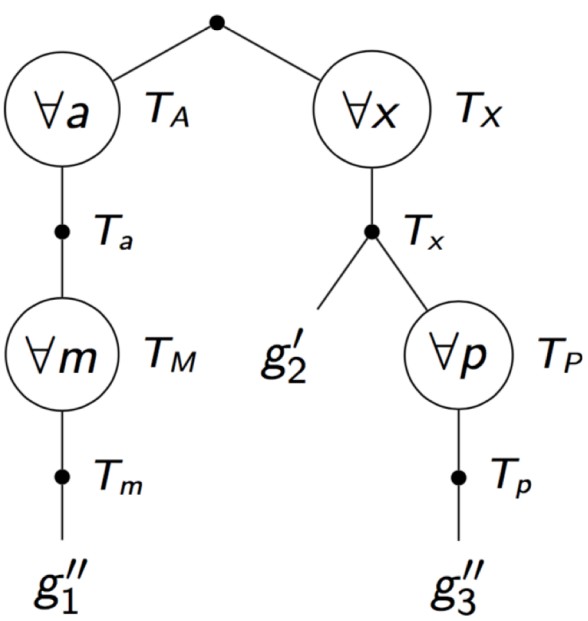
$$context(T) = RV(T) \cap acutset(T), RV(T) = \bigcup_{T' \in child(T)} RV(T'), RV(L) = RV(\phi_L), L \text{ leaf}$$

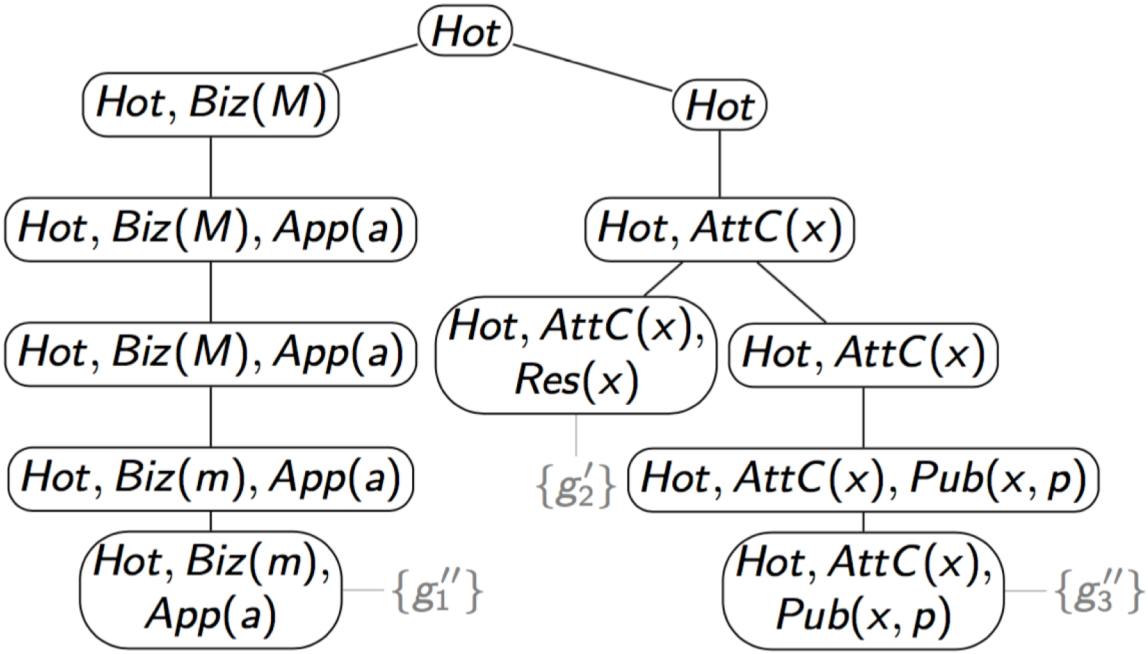
$$cluster(T) = cutset(T) \cup context(T), cluster(L) = RV(\phi_L), L \text{ leaf}$$



# FO Dtree to FO Jtree

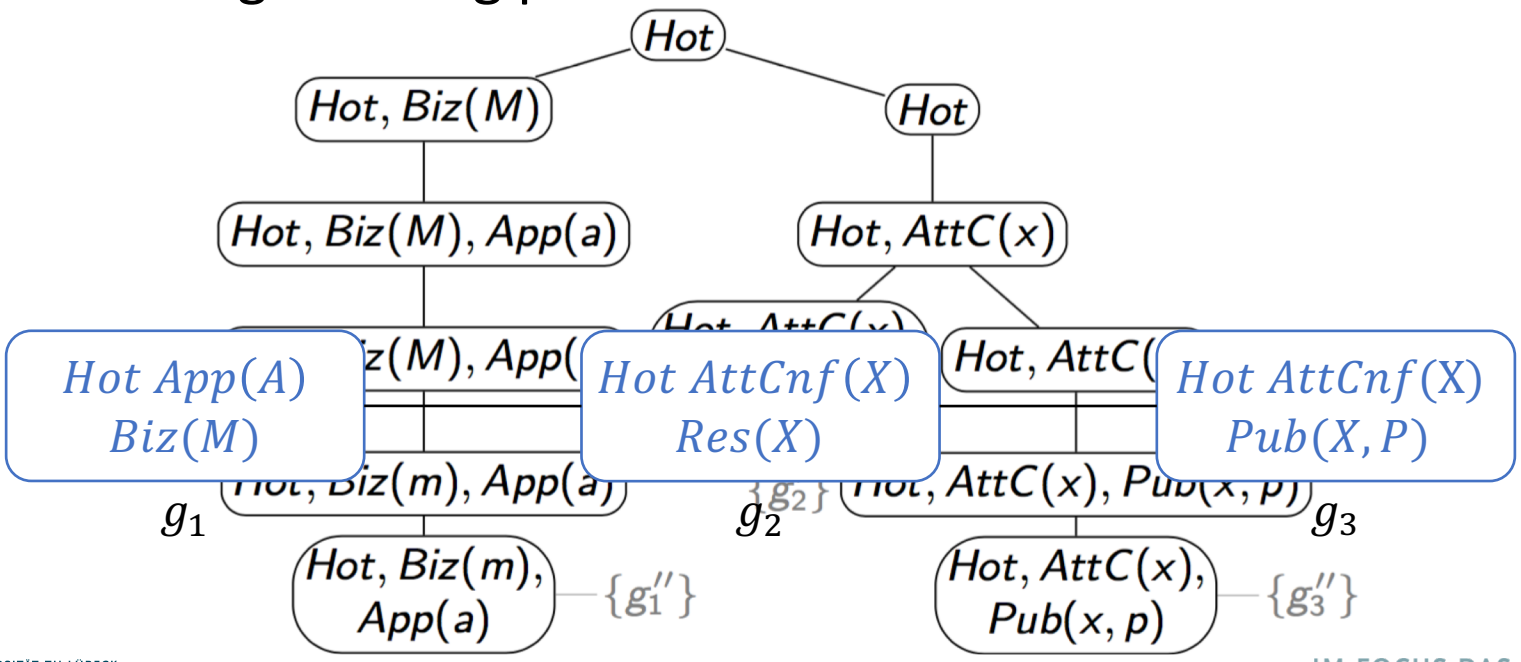
- Compute clusters per node and minimise



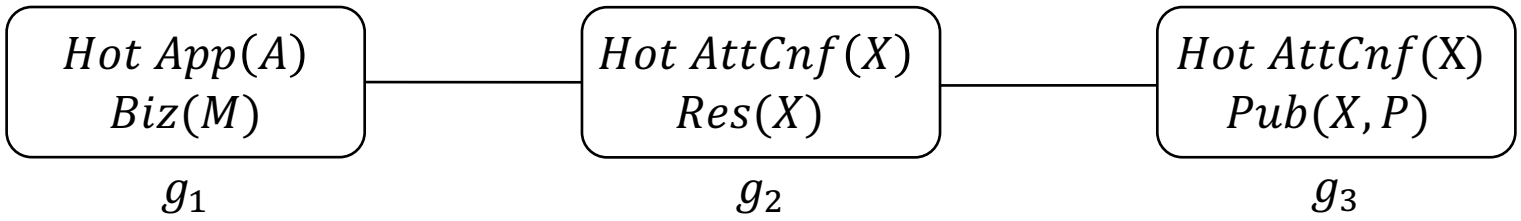
# Minimising an FO Jtree

- FO jtree is minimal
  - If by removing a variable from any parcluster, the FO jtree stops being an FO jtree
- Merge parclusters
  - If neighbouring parclusters are subsets of each other



# Lifted Junction Tree Algorithm: LJT

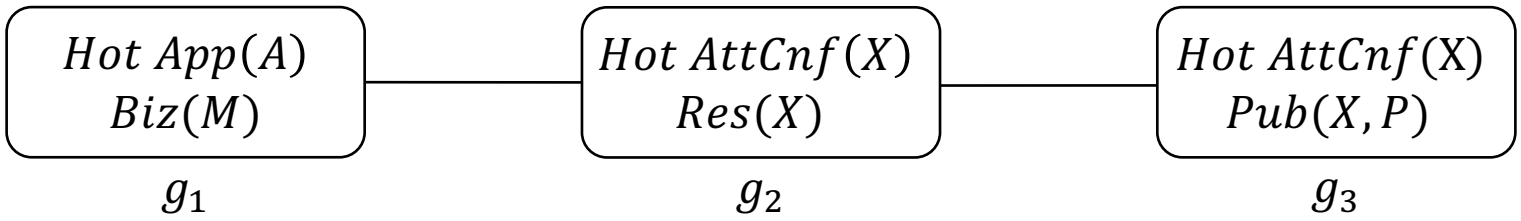
- Input
  - Model $G$
  - Evidence $\boldsymbol{E} = \{AttCnf(alice) = 1, AttCnf(eve) = 1\}$
  - Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
- Algorithm
  1. Build FO jtree $J$ for $G$



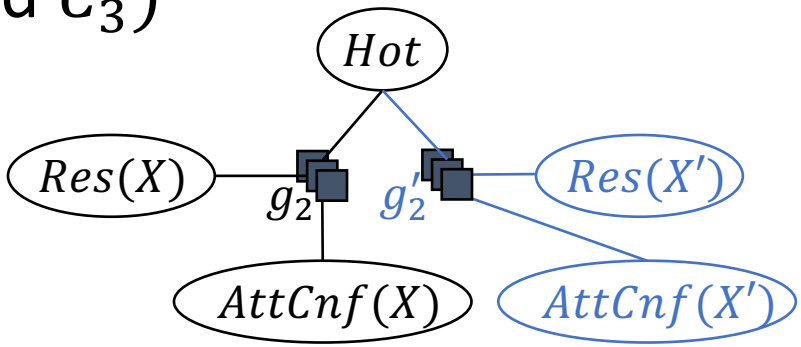

  2. Enter evidence $\boldsymbol{E}$ into $J$

# LJT: Enter Evidence

- At every parcluster that contains evidence variables
  - Evidence $\boldsymbol{E} = \{AttCnf(eve) = 1, AttCnf(alice) = 1\}$
  - Parclusters $C_2$ and $C_3$

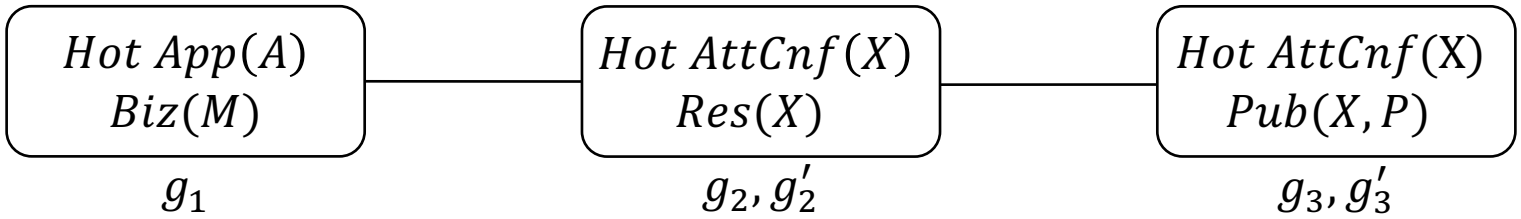



- Enter evidence, at $C_2$ (and $C_3$)
  - Split local model
  - Absorb evidence, in $g_2'$ (possibly exponentiate)



# Lifted Junction Tree Algorithm: LJT

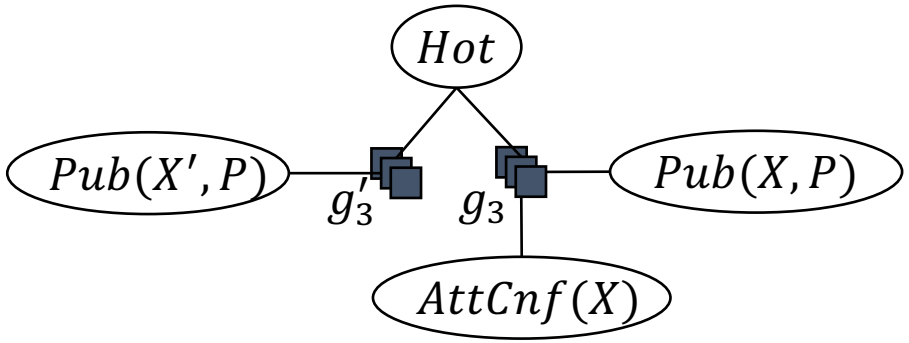
- Input
  - Model $G$
  - Evidence $\boldsymbol{E} = \{AttCnf(alice) = 1, AttCnf(eve) = 1\}$
  - Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
- Algorithm
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $\boldsymbol{E}$ into $J$

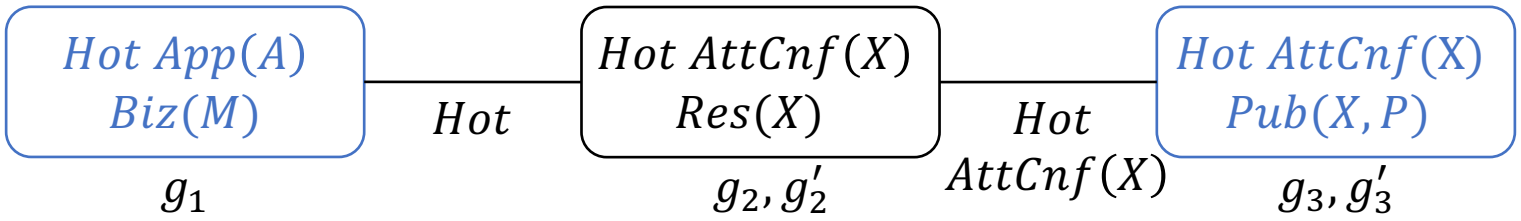



3. Pass messages in $J$

# LJT: Pass Messages

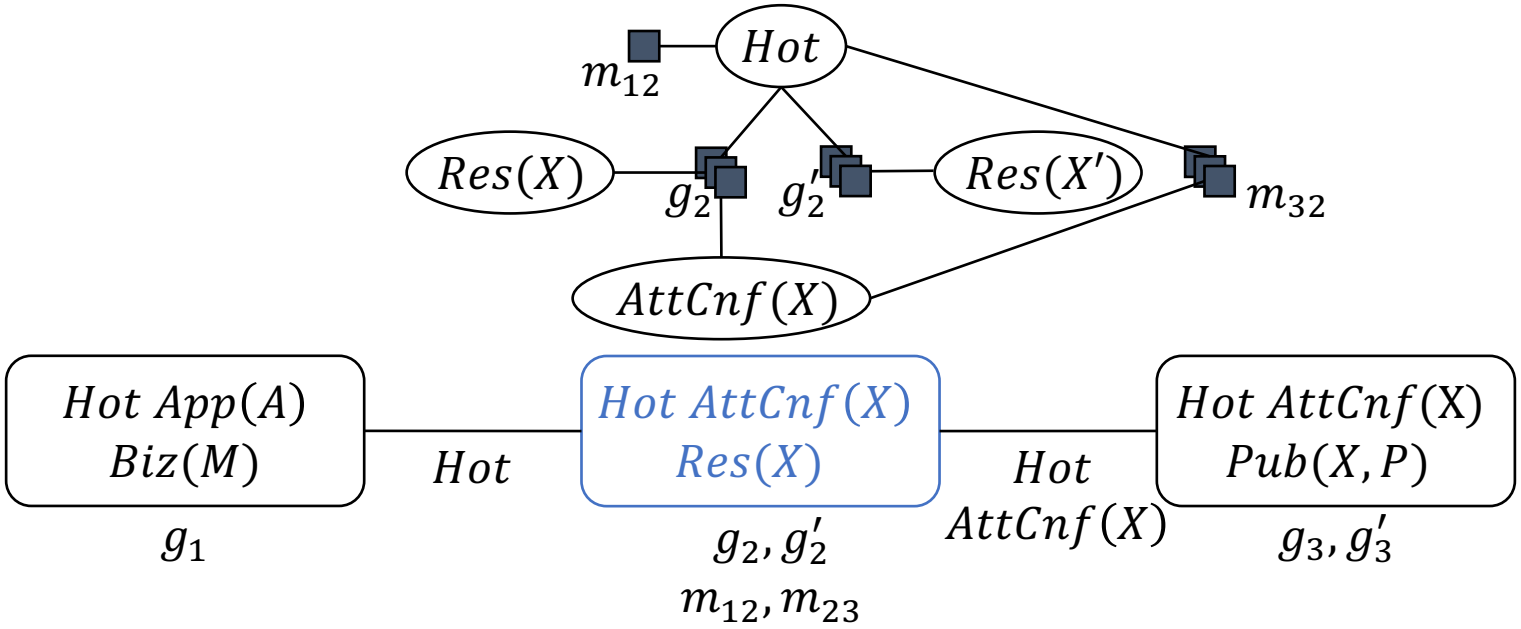
- Inbound
  - $m_{12}$ from $C_1$ to $C_2$ over $Hot$: eliminate $App(A), Biz(M)$
  - $m_{23}$ from $C_3$ to $C_2$ over $Hot, AttCnf(X)$: eliminate $Pub(X, P), Pub(X', P)$



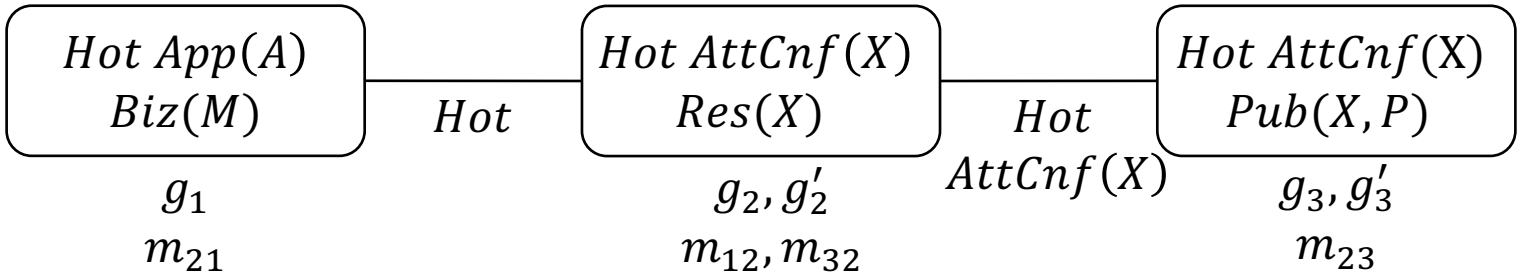
# LJT: Pass Messages

- Outbound
  - $m_{21}$ from $C_2$ to $C_1$ over $Hot$: eliminate $AttCnf(X), Res(X)$ from $g_2, g_2', m_{32}$
  - $m_{32}$ from $C_2$ to $C_3$ over $Hot, AttCnf(X)$: eliminate $Res(X), Res(X')$ from $g_2, g_2', m_{12}$



# Lifted Junction Tree Algorithm: LJT

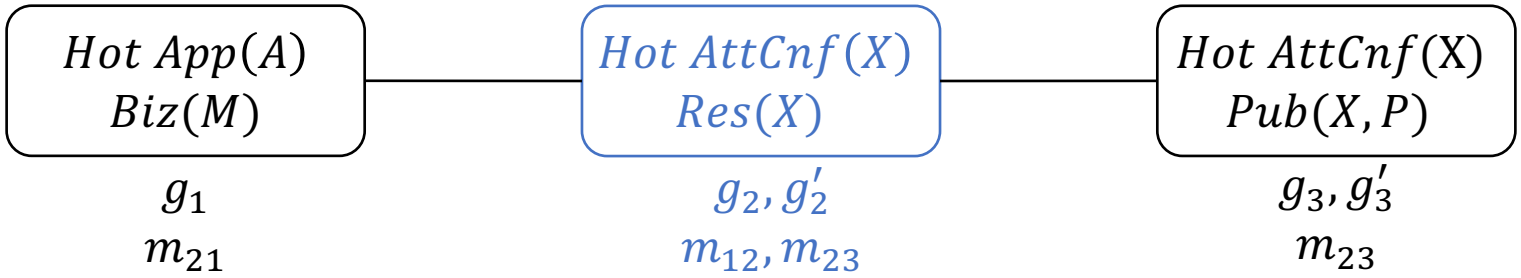
- Input
  - Model $G$
  - Evidence $\boldsymbol{E}$
  - Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
- Algorithm
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $\boldsymbol{E}$ into $J$
  3. Pass messages in $J$




4. Answer queries $\boldsymbol{Q}$

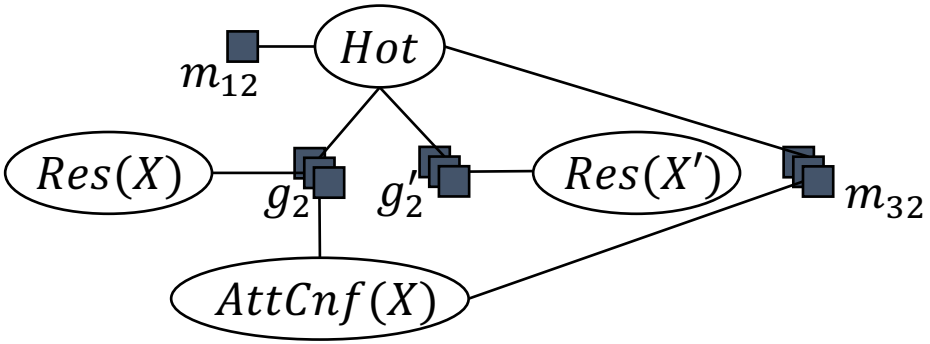
# LJT: Answer Queries

- Find subtree that covers the query variables
- Extract submodel without duplicate information
- Use LVE to answer query
- Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
  - $Q_1 = Res(eve)$



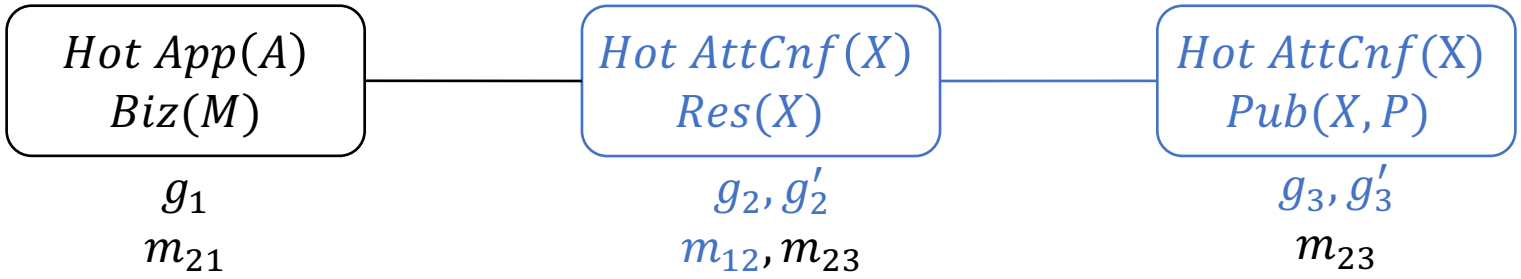
# LJT: Answer Queries

- Find subtree that covers the query variables
- Extract submodel without duplicate information
- Use LVE to answer query
- Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
  - $Q_1 = Res(eve)$
    - Split model
    - Eliminate non-query variables
    - Normalise



# LJT: Answer Queries

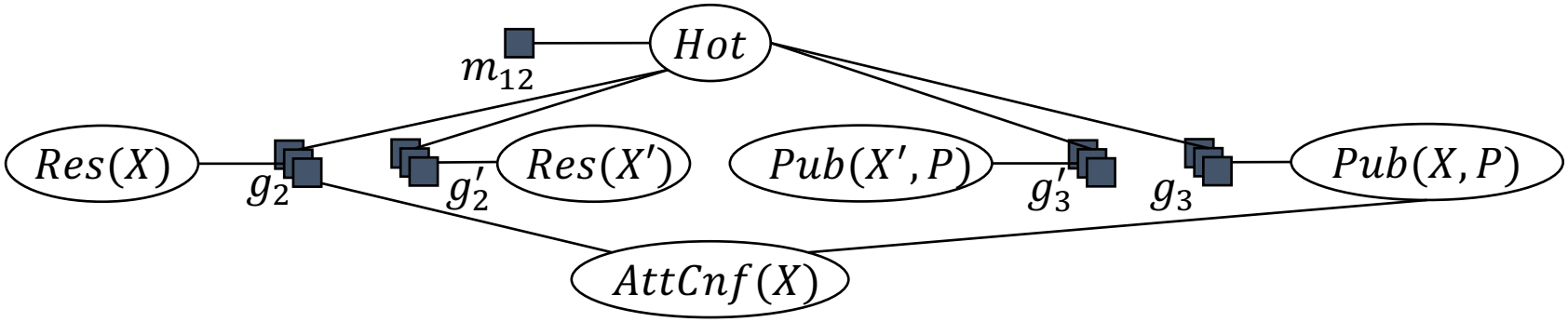
- Find subtree that covers the query variables
- Extract submodel without duplicate information
- Use LVE to answer query
- Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
  - $Q_2 = Res(eve) \wedge Pub(eve, p_1)$



# LJT: Answer Queries

- Find subtree that covers the query variables
- Extract submodel without duplicate information
- Use LVE to answer query
- Queries $\boldsymbol{Q} = \{Res(eve), Res(eve) \wedge Pub(eve, p_1)\}$
  - $Q_2 = Res(eve) \wedge Pub(eve, p_1)$
    - Split model
    - Eliminate non-query variables
    - Normalise

**All under evidence**
$\boldsymbol{E = \{AttCnf(eve) = 1, AttCnf(alice) = 1\}}$

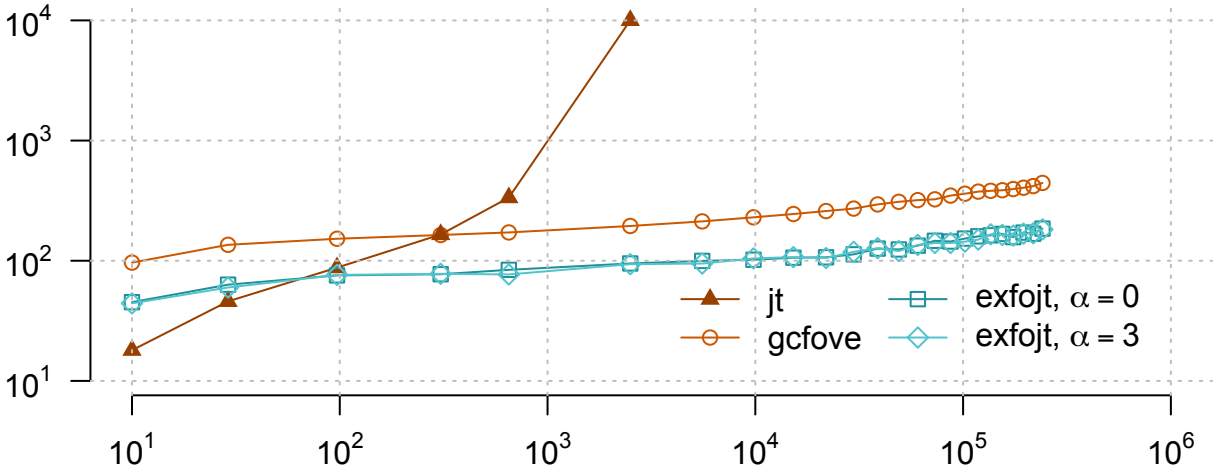


# LJT: Performance

- Static overhead
  - Construction
  - Evidence entering
  - Message passing
- Payoff during QA
  - More space required

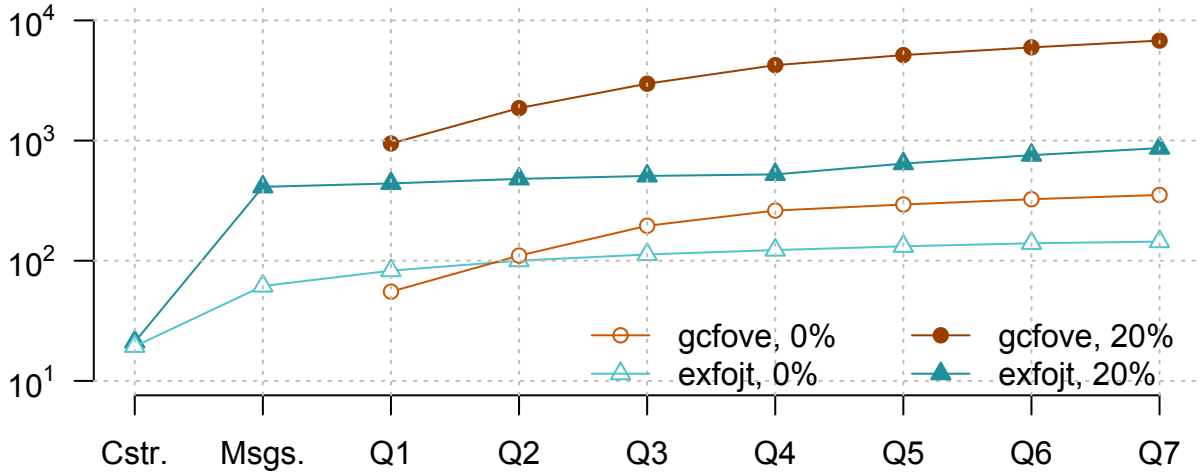
- Prototype implemenation
  - LVE by Taghipour https://dtai.cs.kuleuven.be/software/gcfove
  - LJT

# LJT: Grounded versus Lifted



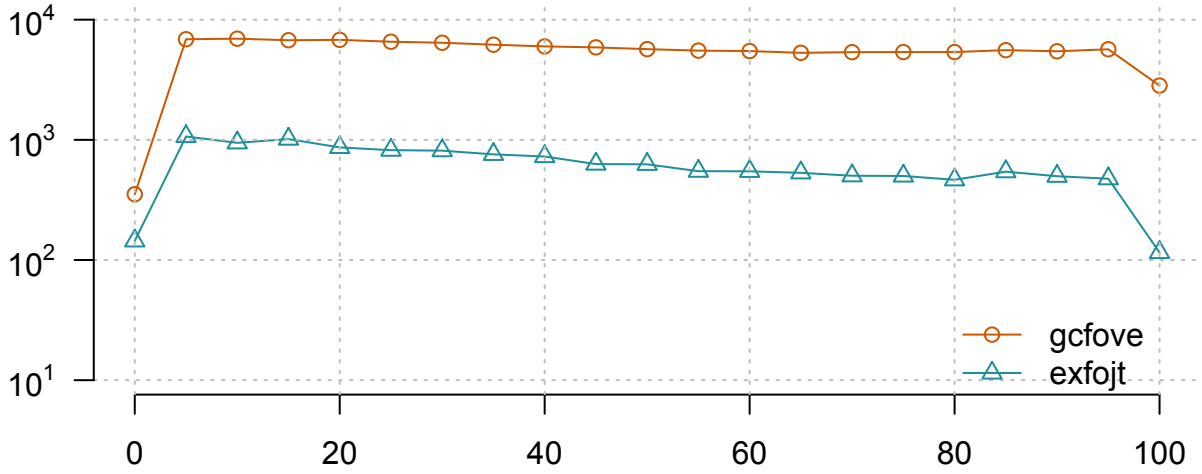

Runtimes in milliseconds , seven queries, grounded model size between 10 and 100,000
Evidence at 0%, $\alpha$ refers to a fusion step not presented here

# LJT: Payoff versus LVE




Accumulated runtimes in milliseconds , seven queries, grounded model size: 100,000
Evidence at 0% and 20% on PRVs with one parameter

# LJT: Evidence

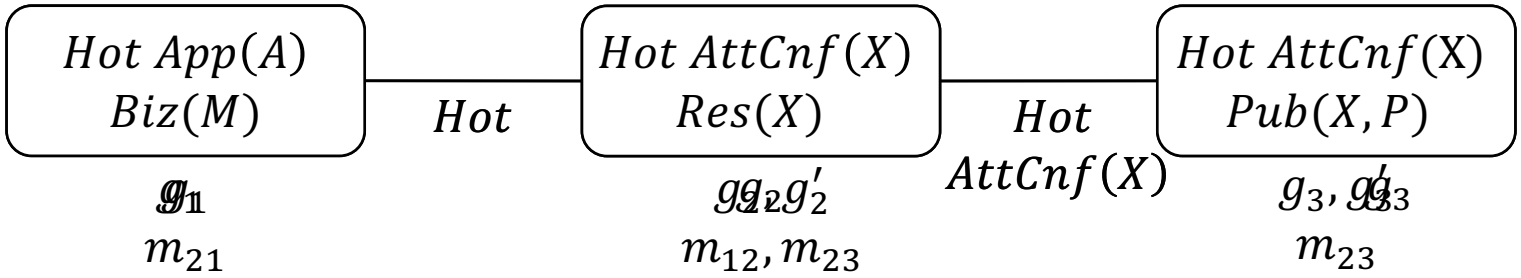



Runtimes in milliseconds , seven queries, grounded model size: 100,000
Evidence from 0% to 100% in 5% steps on PRVs with one parameter

# LJT: New Model/Evidence/Queries

- Changing inputs
  - Queries $\boldsymbol{Q}$
    - Continue at Step 4
  - Evidence $\boldsymbol{E}$
    - Restart at Step 2
  - Model $G$
    - Restart at Step 1

- Algorithm
  1. Build FO jtree $J$ for $G$
  2. Enter evidence $\boldsymbol{E}$ into $J$
  3. Pass messages in $J$
     - Inbound
     - Outbound
  4. Answer queries $\boldsymbol{Q}$



# QA: Most probable assignment

Dawid (1992), Dechter (1999)

- to all variables: most probable explanation (MPE)

$$\underset{rv(G)}{\operatorname{argmax}} P_G$$

  - (L)VE: maxing out (instead of summing out)
  - (L)JT: message calculation by maxing out

- to subset of variables: maximum a posteriori (MAP)

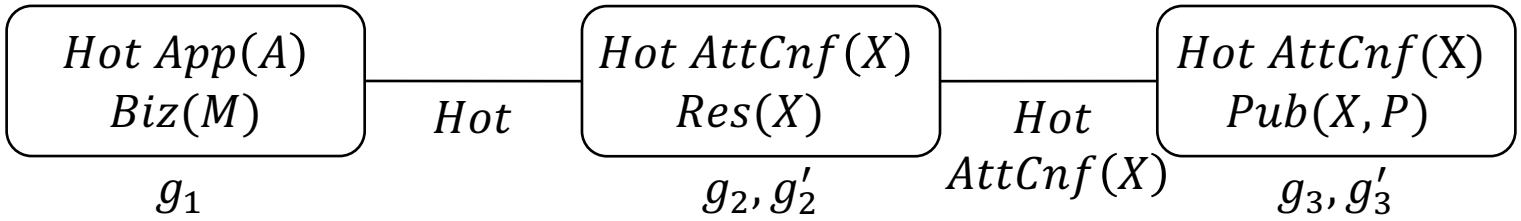
$$\underset{Res.eve,Biz.m_1}{\operatorname{argmax}} \sum_{\substack{r(App(A)),r(Biz(M)),M\neq m_1,\\ r(AttCnf(X)),r(Pub(X,P)),\\ r(Res(X)),X\neq eve,r(Hot)}} P_G$$

  - *MAP a more general case of MPE*
  - $\sum$ **and** $\max$ **not commutative!**

# LJT: MPE

Braun and Möller (2018)

- As before: construction, evidence entering
- Message passing
  - Only one message pass (from periphery to centre)
  - Max out remaining variables at centre



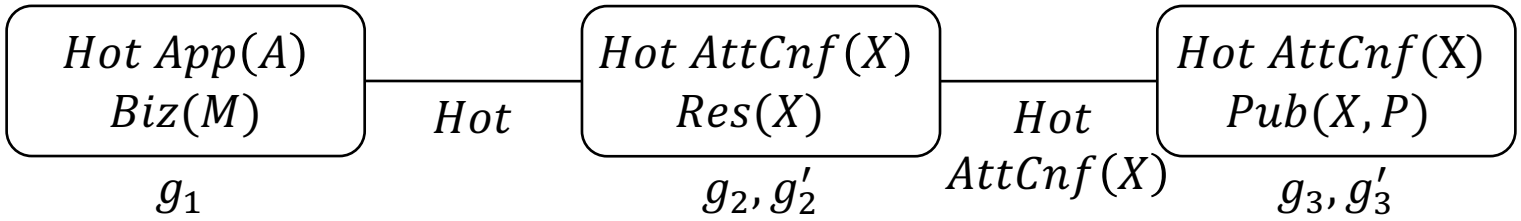
# LJT: MPE

- Answering an MPE
  - $m_{12}$ through maxing out $App(A), Biz(M)$
  - $m_{32}$ through maxing out $Pub(X, P)$
  - At $C_2$, max out $Res(X)$, AttCnf(X)

$$Hot = 0 \quad \forall A, M : App(A) = 0, Biz(M) = 0$$

$$\forall X' \in \{alice, eve\}, P : Res(X') = 1, AttCnf(X') = 1, Pub(X', P) = 1$$

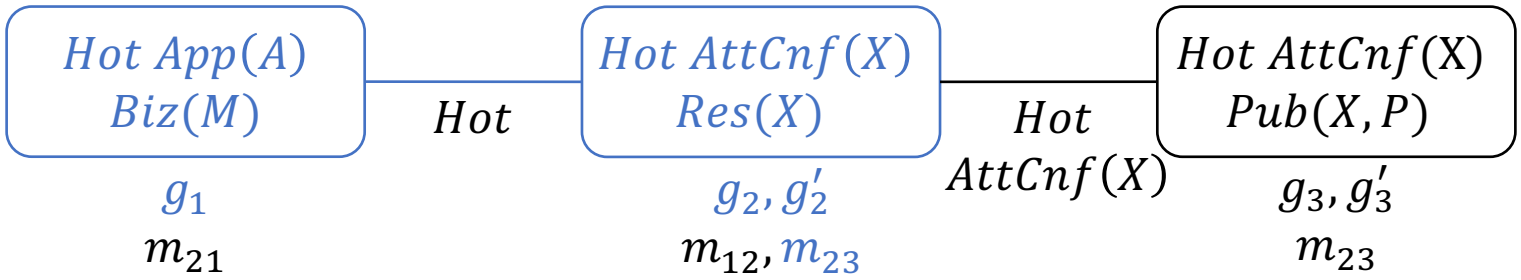
$$\forall X, P : Res(X) = 1, AttCnf(X) = 1, Pub(X, P) = 1$$



# LJT: MAP

Braun and Möller (2018)

- As before: construction, evidence entering, message passing (with summing out)
- Answer MAP query: get submodel (as before)
  - Sum out non-query variables
  - Max out query variables
  - E.g., $Res.eve, Biz.m_1$
    - Sum out $Hot, App(A), AttCnf(X), Res(X), X \neq eve, Biz(M), M \neq m_1$
    - Max out $Res.eve, Biz.m_1$

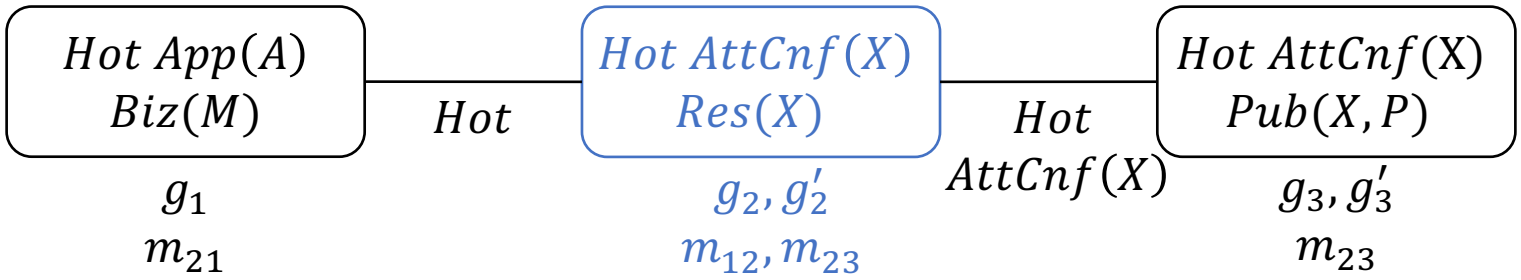


# LJT: MAP

Braun and Möller (2018)

- Assignment to complete parclusters safe
  - After message pass, max out parcluster variables
  - E.g., MAP over $C_2$
    - Max out $AttCnf(X), Res(X), Hot$

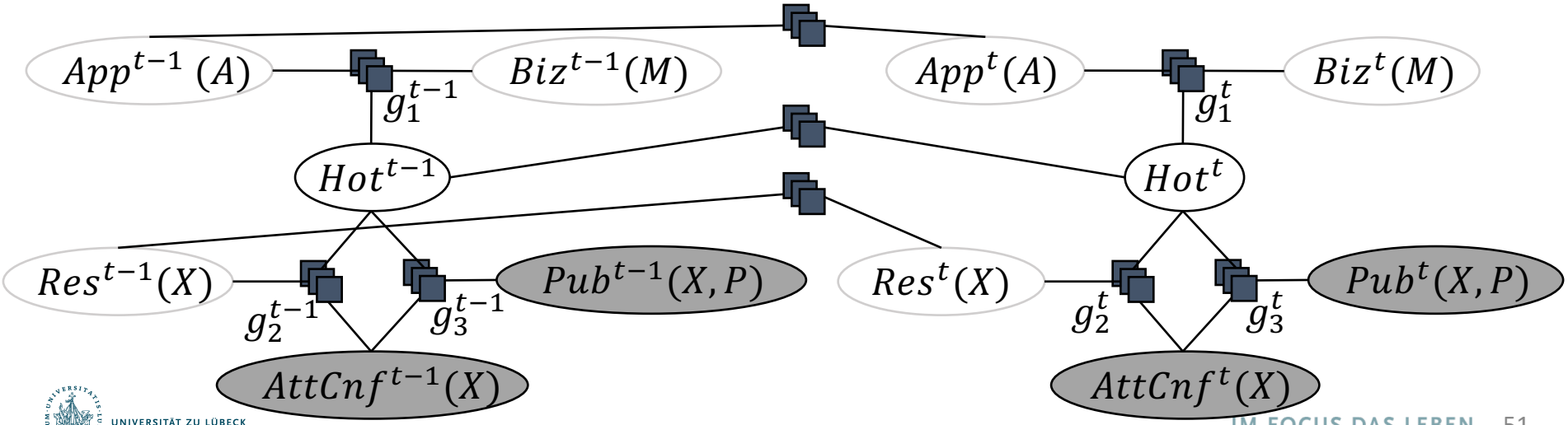
$$Hot = 0$$
$$\forall X' \in \{alice, eve\}: Res(X') = 1, AttCnf(X') = 1$$
$$\forall X : Res(X) = 1, AttCnf(X) = 1$$



# Dynamic Parfactor Graphs

Gehrke et al. (2018)

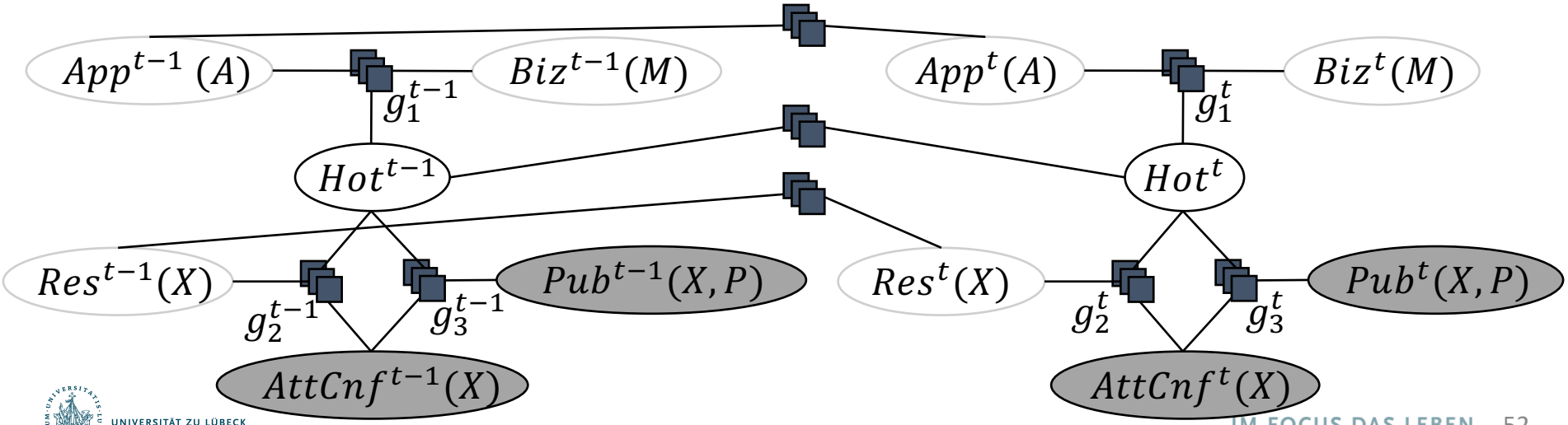
- As before: duplicate model for time slices
  - Connect PRVs from one slice to next
- Inference tasks
  - Filtering $P(X^t|E^{0:t})$, Prediction $P\left(X^{t+k}\middle|E^{0:t}\right)$
  - Smoothing $P\left(X^k\middle|E^{0:t}\right), k < t$
  - MAP, MPE (Viterbi)



# Lifted Dynamic Junction Tree Alg.

Gehrke et al. (2018)

- Build two FO jtrees
  - For model with t=0 (one slice)
  - For 1.5 time-slice model (one slice plus predecessors)
  - Ensure PRVs connected to next slice in one cluster (interface)
- Within time-slice: reasoning with LJT
  - Message to transport current information to next slice



# Outlook

- Optimising LJT
  - Parallelisation
  - Caching
- From discrete over interval to continuous ranges
- Learning?
  - Structure
  - Potentials
  - Symmetries
- Open world?
  - Unknown domains
  - Unknown behaviour

# References


- Dawid (1992)
  Alexander Philip Dawid. Applications of a General Propagation Algorithm for Probabilistic Expert Systems. *Statistics and Computing*, 2(1):25–36, 1992.
- Dechter (1999)
  Rina Dechter. Bucket Elimination: A Unifying Framework for Probabilistic Inference. In *Learning and Inference in Graphical Models*, pages 75–104. MIT Press, 1999.
- Lauritzen and Spiegelhalter (1988)
  Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society. Series B: Methodological*, 50:157–224, 1988.
- Poole and Zhang (2003)
  David Poole and Nevin L. Zhang. Exploiting Contextual Independence in Probabilistic Inference. *Jounal of Artificial Intelligence*, 18:263–313, 2003.

# References


- Shenoy and Shafer (1990)
  Prakash P. Shenoy and Glenn R. Shafer. Axioms for Probability and Belief-Function Propagation. *Uncertainty in Artificial Intelligence 4*, 9:169–198, 1990.
- Taghipour et al. (2013)
  Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. *Journal of Artificial Intelligence Research*, 47(1):393–439, 2013.
- Taghipour et al. (2013a)
  Nima Taghipour, Jesse Davis, and Hendrik Blockeel. First- order decomposition trees. In *Advances in Neural Information Processing Systems 26*, pages 1052–1060. Curran Associates, Inc., 2013.
- Zhang and Poole (1994)
  Nevin L. Zhang and David Poole. A Simple Approach to Bayesian Network Computations. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 171–178, 1994.

# Own Work

- Braun and Möller (2016)
  Tanya Braun and Ralf Möller. Lifted Junction Tree Algorithm. In *Proceedings of KI 2016: Advances in Artificial Intelligence*, pages 30–42, 2016.
- Braun and Möller (2017)
  Tanya Braun and Ralf Möller. Counting and Conjunctive Queries in the Lifted Junction Tree Algorithm. In *Graph Structures for Knowledge Representation and Reasoning - 5th International Workshop, GKR 2017, Melbourne, Australia, August 21, 2017*, 2017.
- Braun and Möller (2017a)
  Tanya Braun and Ralf Möller. Preventing Groundings and Handling Evidence in the Lifted Junction Tree Algorithm. In *Proceedings of KI 2017: Advances in Artificial Intelligence*, pages 85–98, 2017.
- Braun and Möller (2018)
  Tanya Braun and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.

# Own Work

- Gehrke et al. (2018)
  Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *Proceedings of the International Conference on Conceptual Structures*, 2018.