Dynamic StarAI

Answering Static Queries

Tutorial at KI 2019

Tanya Braun, Marcel Gehrke, Ralf Möller
Universität zu Lübeck

Agenda: Dynamic Models and Statistical Relational AI

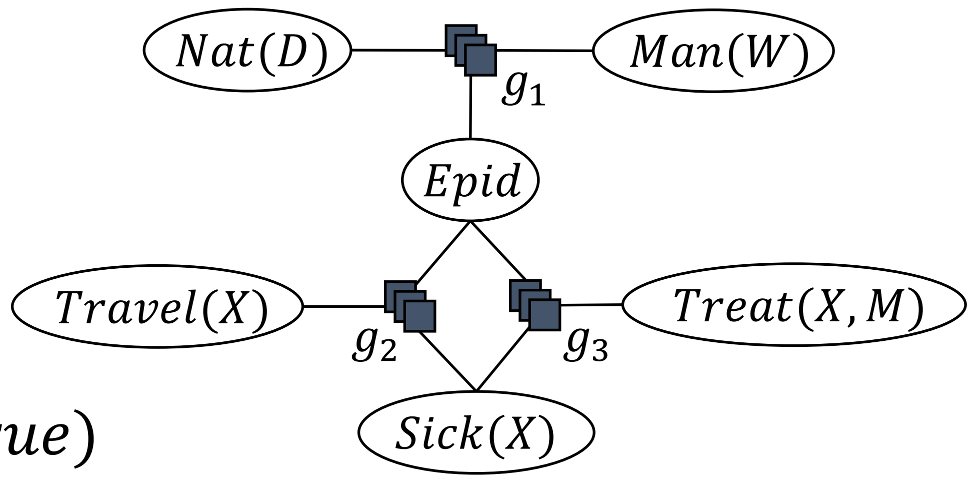
- Probabilistic relational models (PRMs) (Ralf)
- **Answering static queries** (Tanya)
  - **Semantics**
  - **Lifting: Scalable w.r.t. numbers of objects**
  - **Junction Trees: Scalable w.r.t. model size**
- Answering continuous queries (Marcel)
  - Lifted Dynamic Junction Tree Algorithm (LDJT)
  - Relational interfaces
  - Taming reasoning w.r.t. lots of evidence over time
- Take home messages (Ralf)
  - LJT and LDJT research
    relevant for all variants of PRMs

Goal:
Overview of
central ideas

Query Answering (QA): Queries

Avoid groundings!

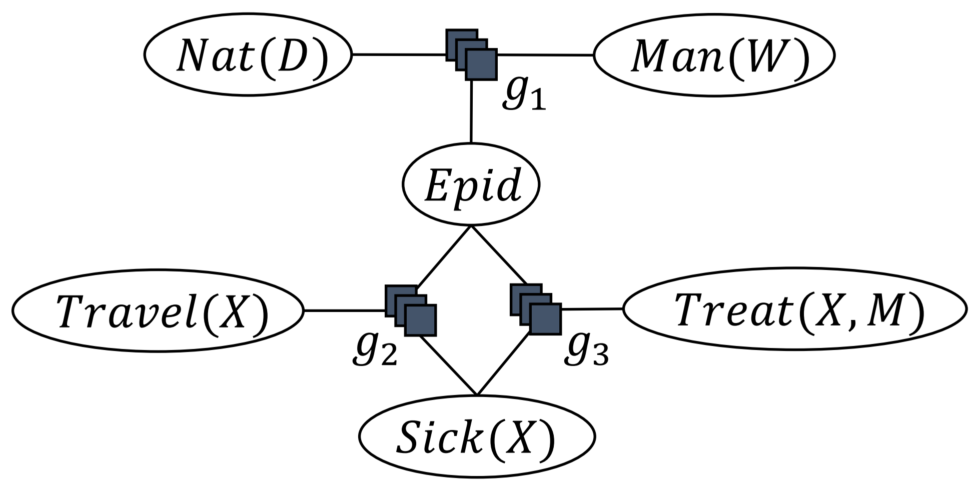
- Marginal distribution
  - $P(Sick(eve))$
  - $P(Travel(eve), Treat(eve, m_1))$
- Conditional distribution
  - $P(Sick(eve) | Epid)$
  - $P(Epid | Sick(eve) = true)$
- Most probable assignment
  (not part of this tutorial)



QA: Lifted Variable Elimination (LVE)

Poole (2003), de Salvo Braz et al. (2005, 2006), Milch et al. (2008), Taghipour et al. (2013, 2013a)

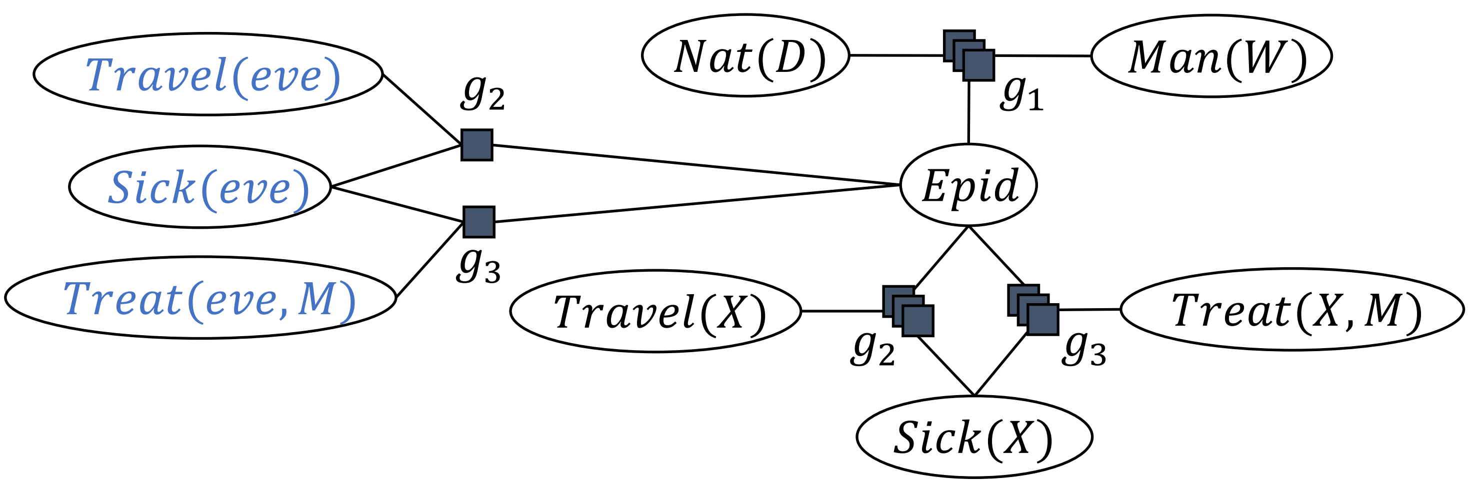
- Eliminate all variables not appearing in query
- Lifted summing out
  1. Sum out representative instance as in propositional variable elimination
  2. Exponentiate result for isomorphic instances



Avoid groundings!

QA: LVE in Detail

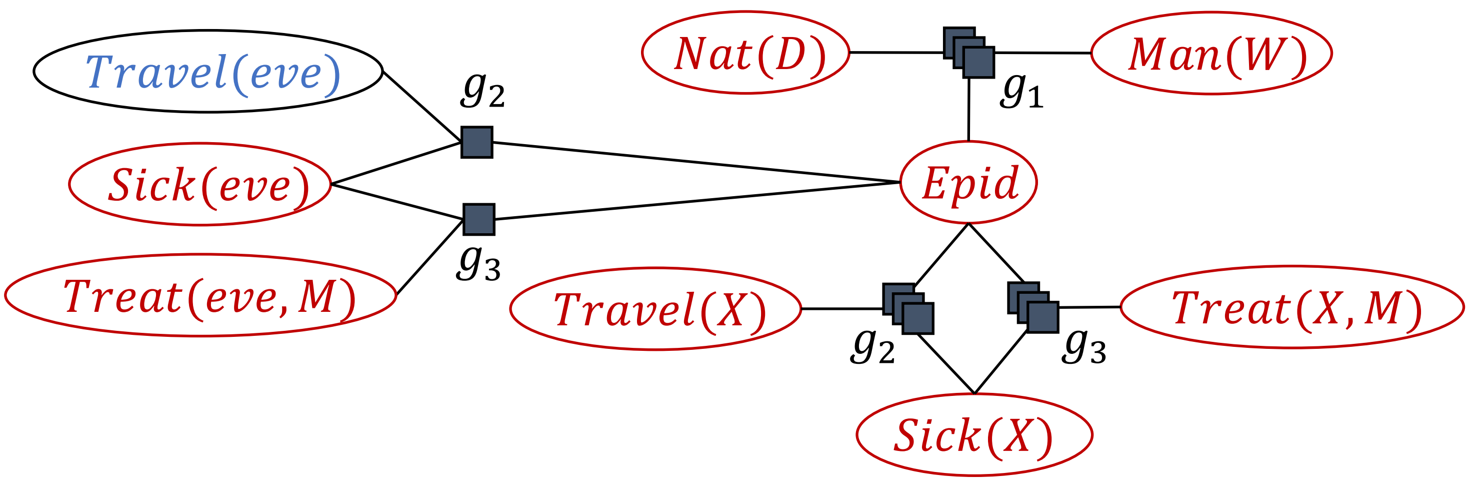
- E.g., marginal
  - $P(Travel(eve))$
  - Split w.r.t. $Travel(eve)$ (each X preemptively)



UNIVERSITÄT ZU LÜBECK

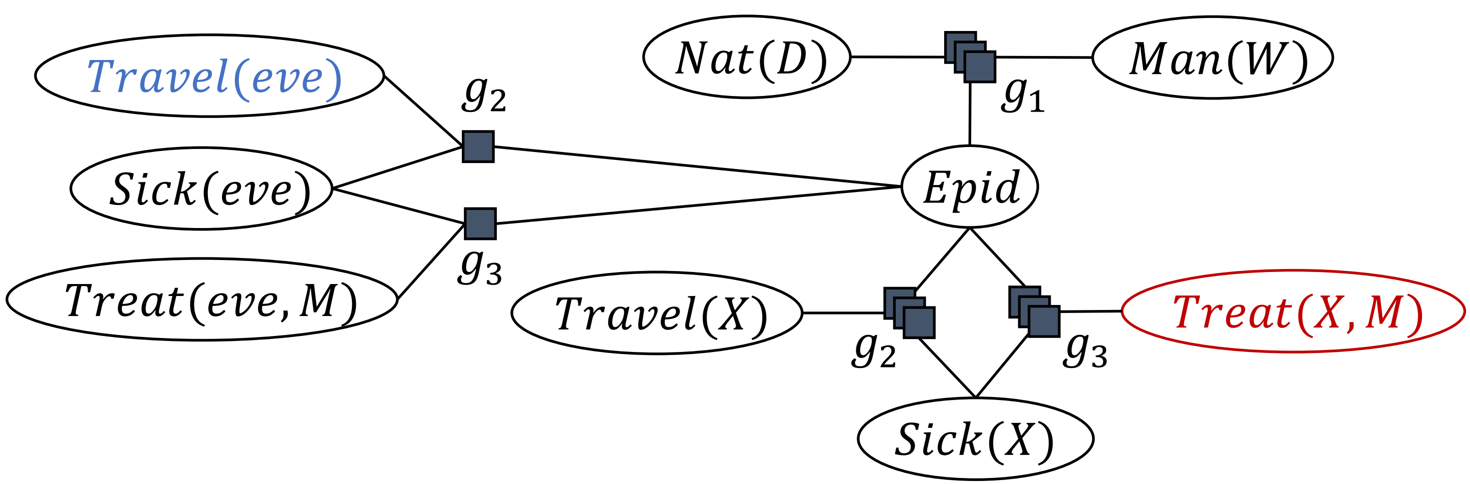
QA: LVE in Detail

- E.g., marginal
  - $P(Travel(eve))$
  - Split w.r.t. $Travel(eve)$ (each X preemptively)
  - Eliminate all non-query variables
  - Normalise



QA: LVE in Detail

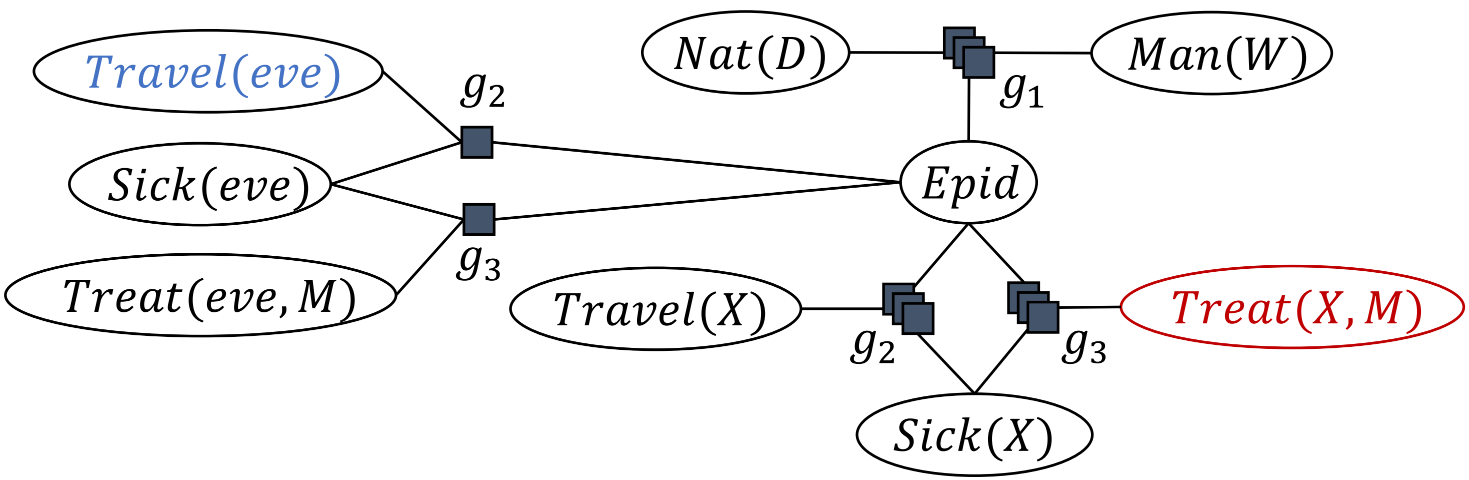
- Eliminate $Treat(X, M)$
  - Appears in only one g: g_3
  - Contains all logical variables of g_3: X, M
  - For each X constant: the same number of M constants
  - ✓Preconditions of lifted summing out fulfilled, lifted summing out possible



LVE in Detail: Lifted Summing Out

- Eliminate $Treat(X, M)$ by lifted summing out
  1. Sum out representative
  2. Exponentiate for indistinguishable objects

$$\left(\sum_{t \in r(Treat(X,M))} g_3(Epid = e, Sick(X) = s, Treat(X, M) = t) \right)^{|M|}$$



LVE in Detail: Lifted Summing Out

- Eliminate $Treat(X, M)$
  1. Sum out representative
  2. Exponentiate for indistinguishable objects

$$\left(\sum_{t \in r(Treat(X,M))} g_3(e, s, t)\right)^{|M|}$$

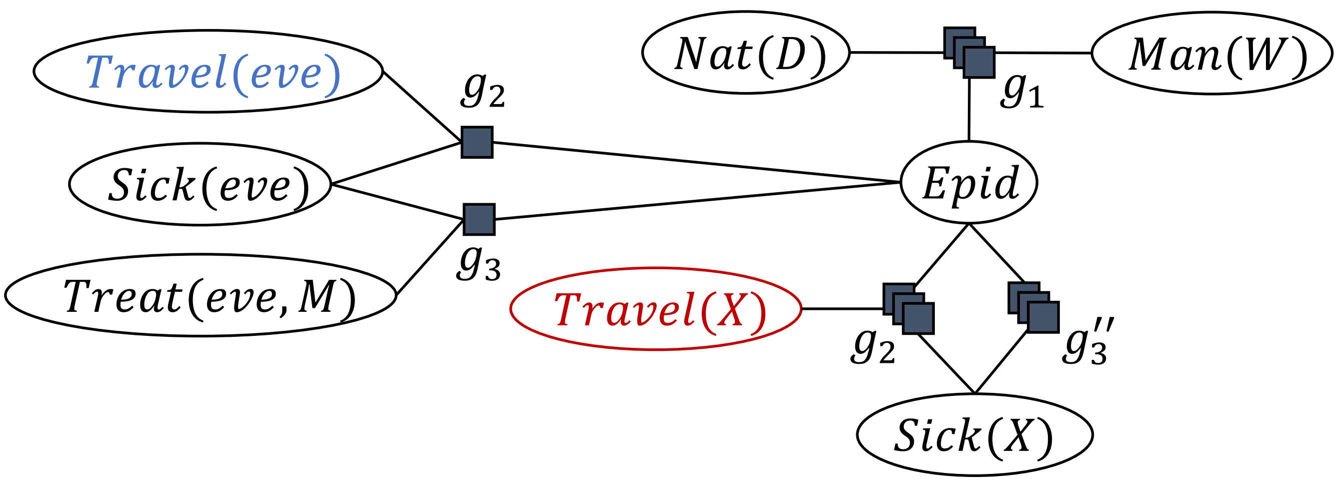
$Epid$	$Sick(X)$	$Treat(X, M)$	g_3
$false$	$false$	$false$	5
$false$	$false$	$true$	1
$false$	$true$	$false$	3
$false$	$true$	$true$	2
$true$	$false$	$false$	5
$true$	$false$	$true$	4
$true$	$true$	$false$	1
$true$	$true$	$true$	7

+ + + +

$Epid$	$Sick(X)$	g_3'	g_3''
$false$	$false$	6	$6^2 = 36$
$false$	$true$	5	$5^2 = 25$
$true$	$false$	9	$9^2 = 81$
$true$	$true$	8	$8^2 = 64$

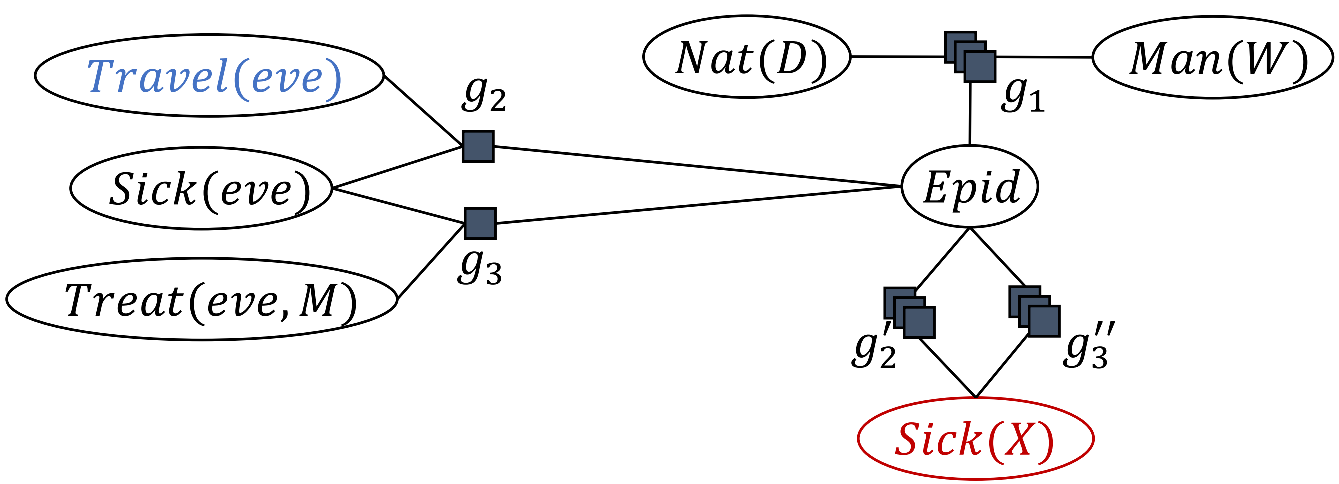
QA: LVE in Detail

- After eliminating $Treat(X, M)$
  - Eliminate $Travel(X)$
    - Does not eliminate logical variable (unlike M)
    - Yields $g_2'\big(Epid, Sick(X)\big)$



QA: LVE in Detail

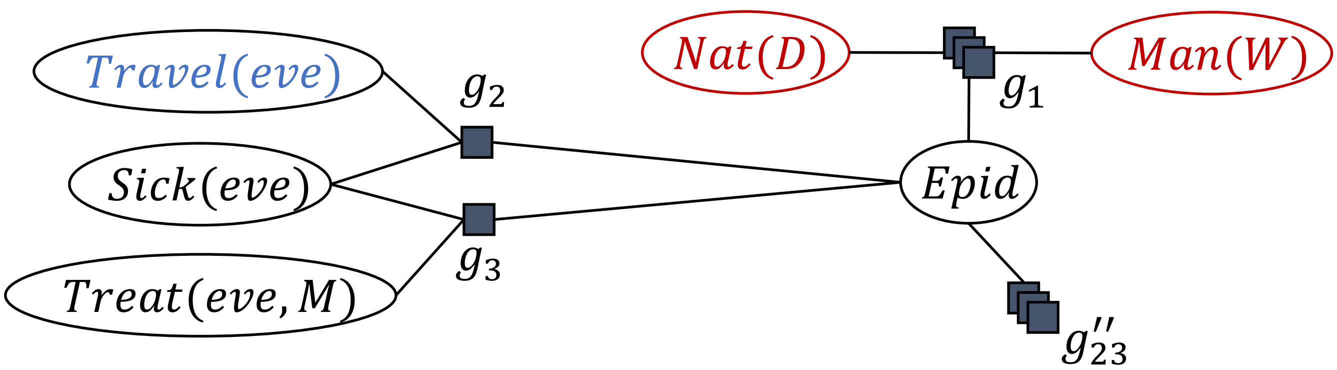
- After eliminating $Treat(X, M), Travel(X)$
  - Eliminate $Sick(X)$
    - Requires multiplication of g_2' and g_3''
    - Eliminates X
    - Yields g_{23}''



UNIVERSITÄT ZU LÜBECK

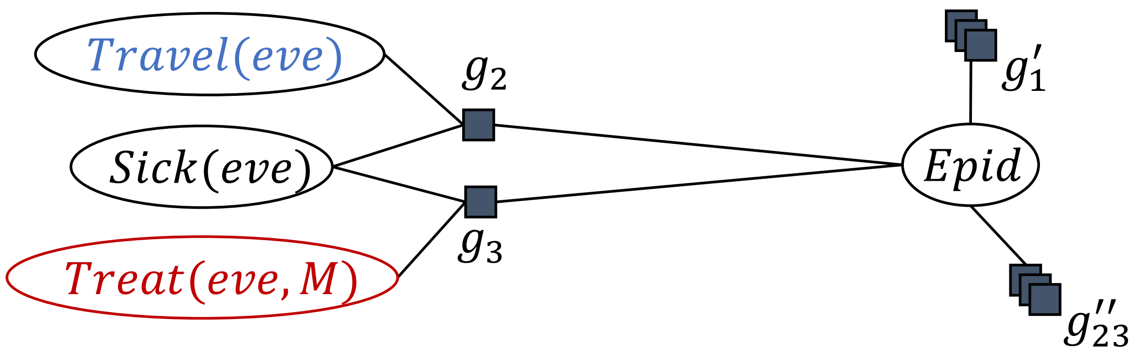
QA: LVE in Detail

- After eliminating $Treat(X, M), Travel(X), Sick(X)$
  - Problem in g_1: No PRV contains all logical variables of g_1
    - $Nat(D)$ does not contain W, $Man(W)$ does not contain D
  - Requires count conversion of g_2' and g_3''
    - Counts logical variables given preconditions (Milch et al. 2008)
    - Counting D enables lifted summing out of $Man(W)$, then summing out of count converted $Nat(D)$



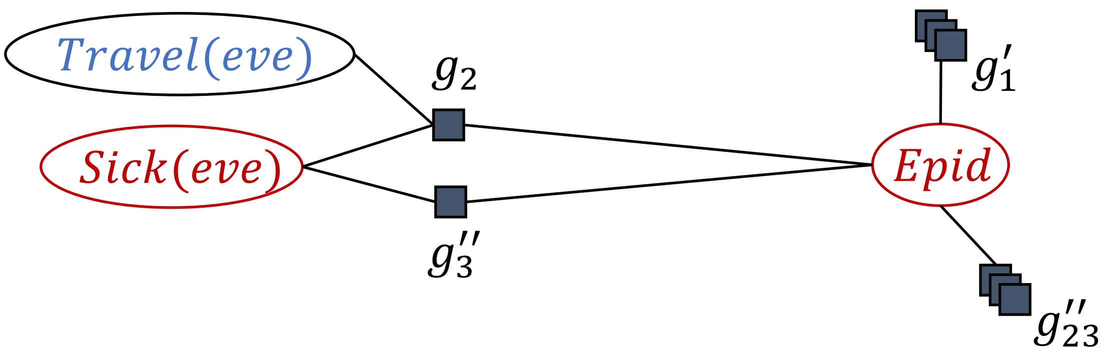
QA: LVE in Detail

- After eliminating $Treat(X, M), Travel(X), Sick(X), Man(W), Nat(D)$
  - Eliminate $Treat(eve, M)$
    - Sum out representative of M, exponentiate result to $|M|$
    - Eliminates last logical variable in remaining model



QA: LVE in Detail

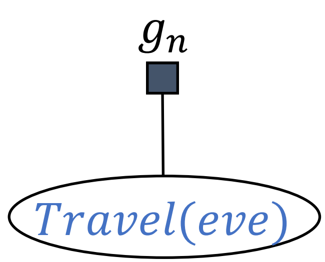
- After eliminating $Treat(X, M), Travel(X), Sick(X), Man(W), Nat(D),$
  $$Treat(eve, M)$$
  - Remaining operations on propositional level
    - Eliminate $Sick(eve)$ after multiplication
    - Eliminate $Epid$ after multiplication



UNIVERSITÄT ZU LÜBECK

QA: LVE in Detail

- After eliminating $Treat(X, M), Travel(X), Sick(X), Man(W), Nat(D),$
  $Treat(eve, M), Sick(eve), Epid$
  - Normalise the final parfactor

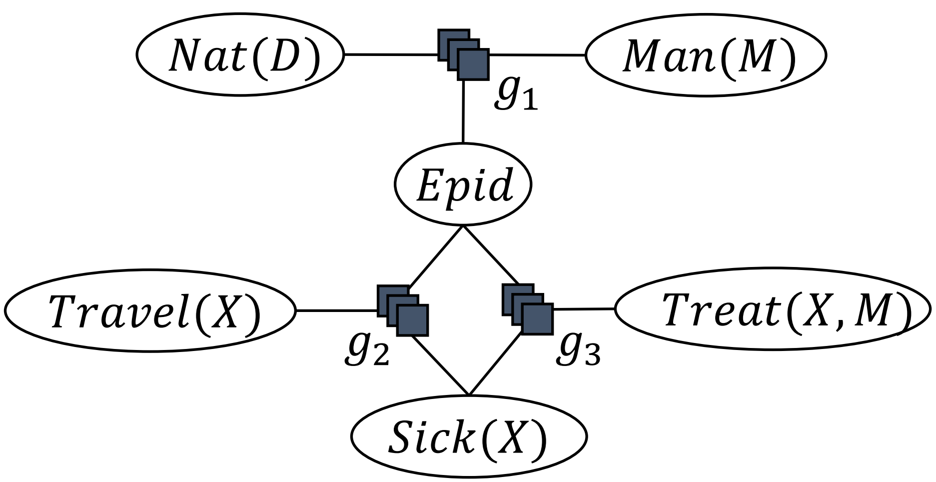


$Travel(eve)$	g
$false$	190
$true$	297

$Travel(eve)$	g_n
$false$	0.39
$true$	0.61

Problem: Many Queries

- Set of queries
  - $P(Travel(eve))$
  - $P(Sick(bob))$
  - $P(Treat(eve, m_1))$
  - $P(Epid)$
  - $P(Nat(flood))$
  - $P(Man(virus))$
  - Combinations of variables
- Under evidence
  - $Sick(X') = true$
  - $\mathcal{D}(X') = \{alice, eve\}$

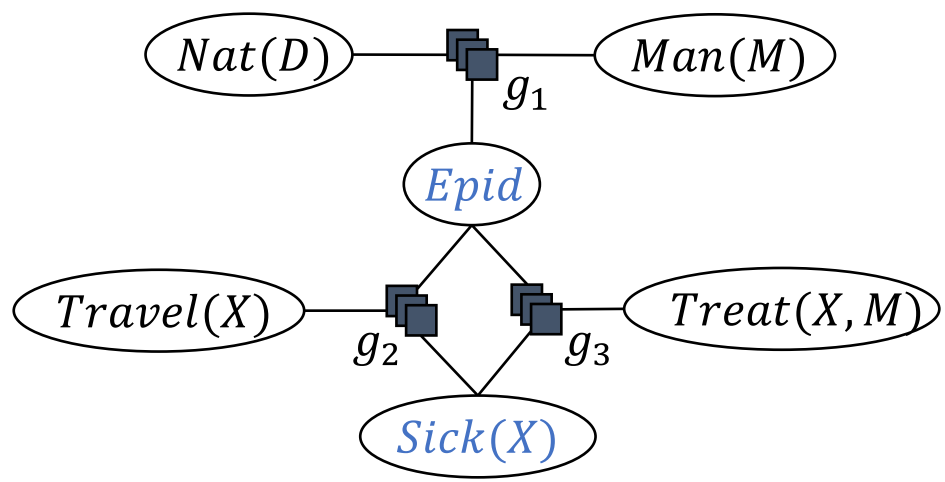


- (L)VE starts with complete model for QA

Solution: Submodels

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990), Braun and Möller (2016)

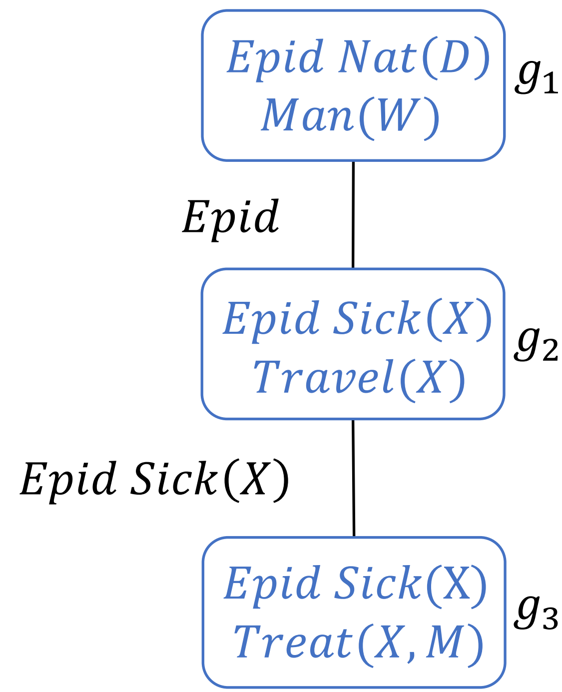
- Identify submodel sufficient for query
  - Find PRVs that make submodel independent from remaining model
    - Separator
  - “Query” over separator collects all influences of remaining model on PRVs in submodel
    - PRVs of submodel = parcluster



Solution: Submodels

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990), Braun and Möller (2016)

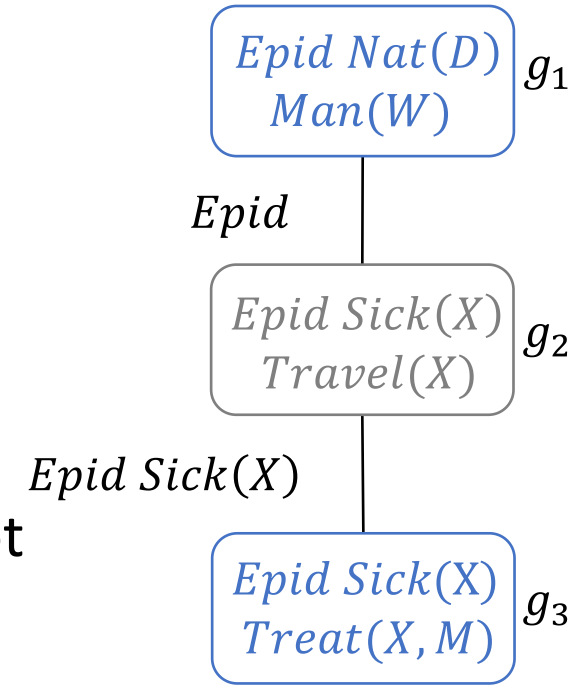
- Network of submodels with separators
  - Recursive "queries" to make submodels independent from each other
  - (First-order) Junction tree
    - DAG, running intersection property
- Recursive queries from each node
  - Arrange queries using dynamic programming
    - Also known as message passing



Message Passing

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990), Braun and Möller (2016)

- Recursive queries arranged in message passes
  1. If a parcluster received messages from all neighbours but one, it sends message to remaining neighbour
     - Automatically true at leaves
     - From periphery to centre (inbound)
  2. If a parcluster received all messages, it sends messages to all neighbours that have not received a message yet
     - First true at some central node
     - And back (outbound)

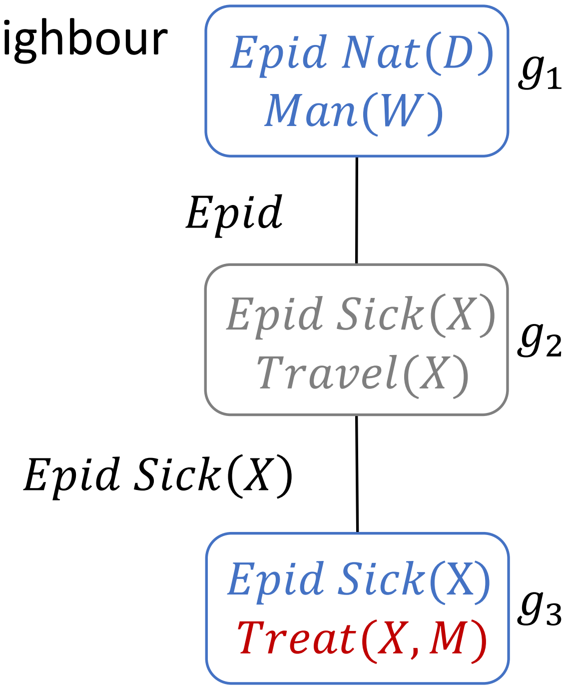


Messages

- Message: Eliminate non-separator variables with LVE
  - E.g., parcluster with g_3
    - Lifted summing out of $Treat(X, M)$
    - Send result as message m_{32} to neighbour

$Epid$	$Sick(X)$	$Treat(X, M)$	g_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

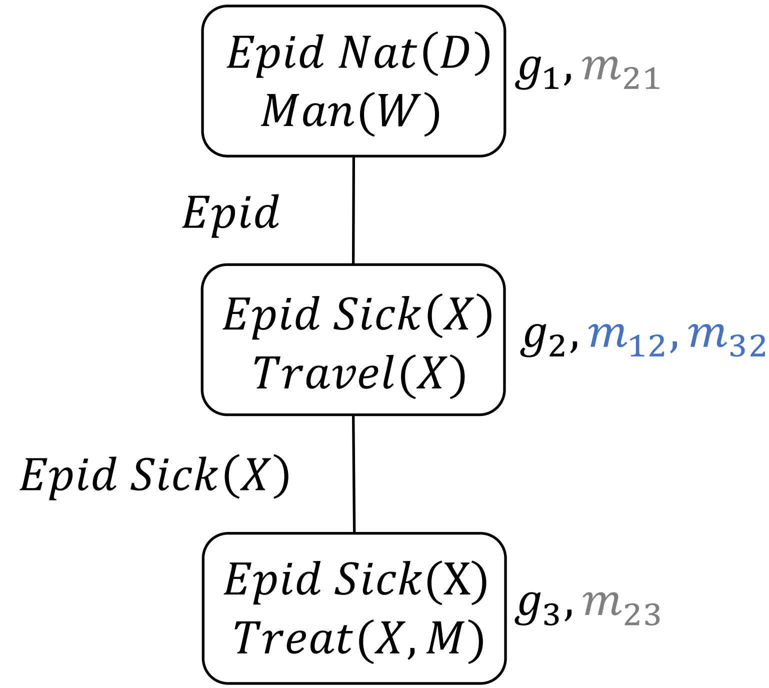
g_3''
$6^2 = 36$
$5^2 = 25$
$9^2 = 81$
$8^2 = 64$



Query Answering in Junction Trees

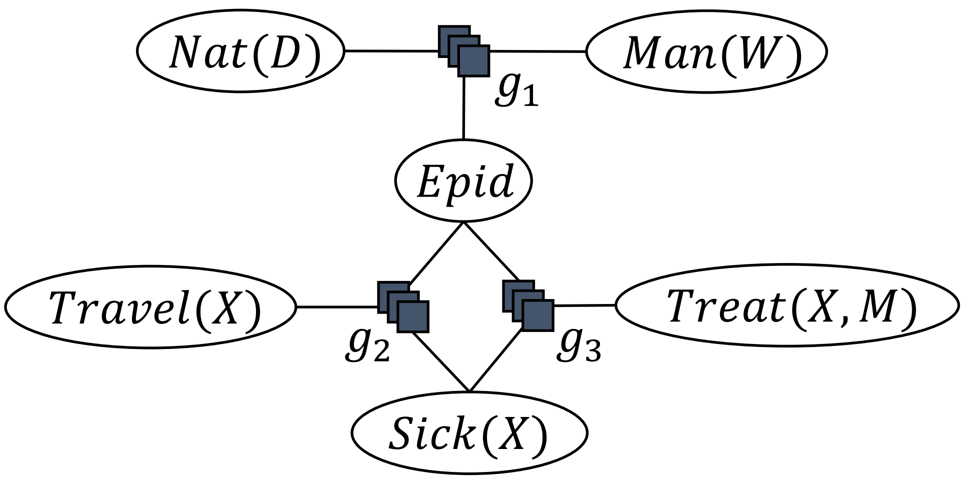
- After two-pass message passing, prepared for any query
- E.g., marginal
  - $P(Travel(eve))$
- Find cluster containing query term
  - Take local model and messages
  - Split w.r.t. $Travel(eve)$
  - Eliminate all non-query variables with LVE
  - Normalise

Lauritzen and Spiegelhalter (1988), Shafer and Shenoy (1989), Jensen et al. (1990), Braun and Möller (2016)

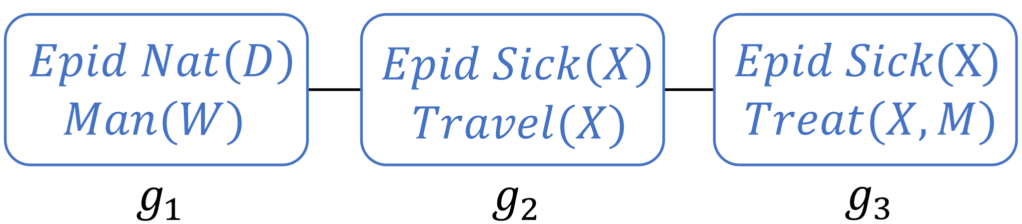


Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017)



Queries on grounded PRVs, e.g.,
$Travel(eve), Treat(eve, m_1), Epid$



- 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 $G = \{g_i\}_{i=1}^{3}$
  - $g_1\big(Epid, Nat(D), Man(W)\big)$
  - $g_2(Travel(X), Epid, Sick(X)\ )$
  - $g_3\big(Epid, Sick(X), Treat(X, M)\big)$

  → Including function specification
- Evidence $\boldsymbol{E} = \{Sick(alice) = true, Sick(eve) = true\}$
- Queries $\boldsymbol{Q} = \{Travel(eve), Epid\}$
- Algorithm
  1. Build FO jtree J for G

UNIVERSITÄT ZU LÜBECK

FO Jtree Construction

- Propositional junction tree construction
  - Triangulation, compute maximum spanning tree, …
  - Hypergraph partitioning
  - Decomposition tree (dtree), clusters, …
- First-order: logical variables

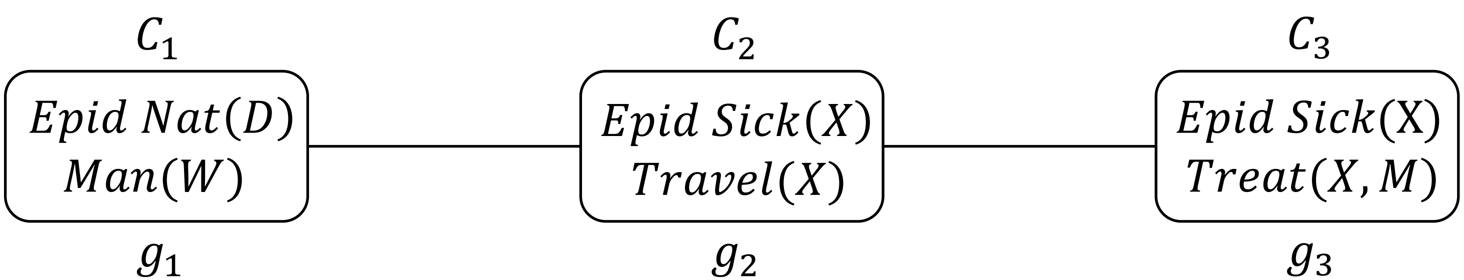
Taghipour et al. (2013b)

  - First-order decomposition trees (FO dtrees)
  - FO dtrees have node properties (cutset, context, cluster)
  - (FO) dtree + clusters = (FO) jtree
  - Heuristic to build an FO dtree
    (logical variables guide the construction)

Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017)

- Input
  - Model G
  - Evidence $\boldsymbol{E}$
  - Queries $\boldsymbol{Q}$
- Algorithm
  1. Build FO jtree J for G



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

LJT: Enter Evidence

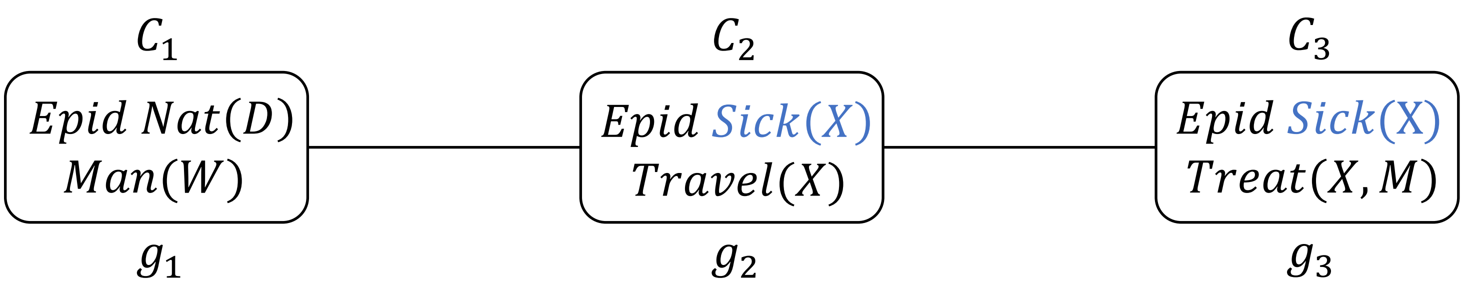
- Evidence as a set of events
  - $\boldsymbol{E} = \{Sick(eve) = true, Sick(alice) = true\}$
- Evidence as a parfactor
  - $g_E\big(Sick(X')\big)$
  - $\mathcal{D}(X') = \{eve, alice\}$
  - Function specification

$Sick(X')$	g_E
$false$	0
$true$	1

- At every parcluster that contains evidence variables, enter evidence

LJT: Enter Evidence

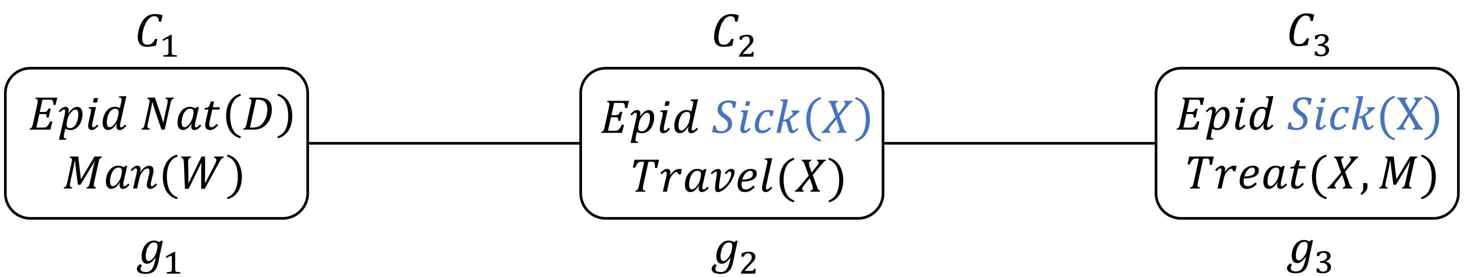
- At every parcluster that contains evidence variables
  - $g_E\big(Sick(X')\big), \mathcal{D}(X') = \{eve, alice\}$
  - Parclusters
    - $Sick(X') \not\subseteq C_1$
    - $Sick(X') \subset C_2$
    - $Sick(X') \subset C_3$



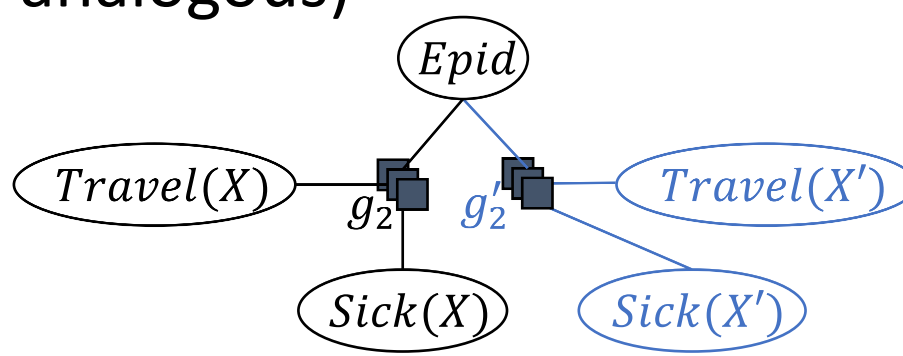
- Enter evidence at C_2 and C_3

LJT: Enter Evidence

- At every parcluster that contains evidence variables
  - $g_E\big(Sick(X')\big), \mathcal{D}(X') = \{eve, alice\}$
  - Parclusters C_2 and C_3



- Enter evidence at C_2 (C_3 analogous)
  - Split local model
    - $\mathcal{D}(X) = \{bob, \ldots\}$
  - Absorb evidence in g_2'



Evidence Absorption

- Absorb $Sick(X') = true$ in g_2'
  - Set values to 0 where $Sick(X') \neq true$
  - Possibly eliminate variable
    - Drop lines with values set to 0
    - Drop column of evidence PRV

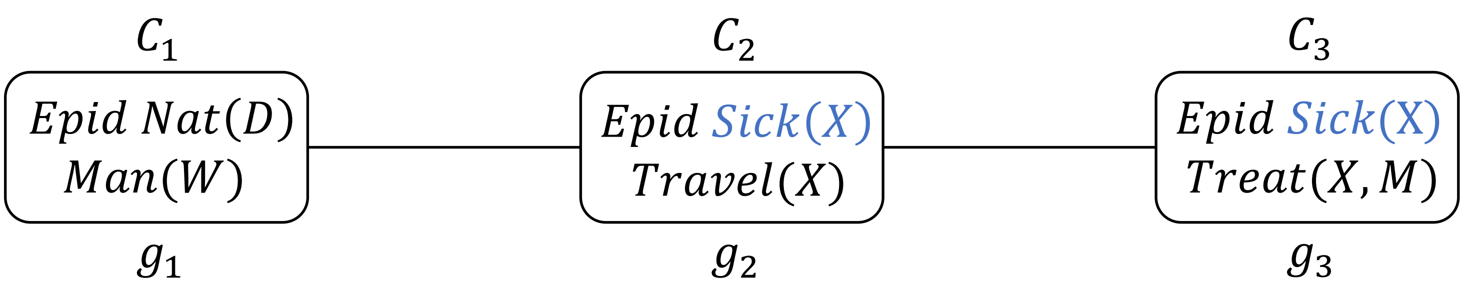
$Travel(X')$	$Epid$	$Sick(X')$	g_2'
$false$	$false$	$false$	$\cancel{5}\ 0$
$false$	$false$	$true$	1
$false$	$true$	$false$	$\cancel{4}\ 0$
$false$	$true$	$true$	6
$true$	$false$	$false$	$\cancel{4}\ 0$
$true$	$false$	$true$	6
$true$	$true$	$false$	$\cancel{2}\ 0$
$true$	$true$	$true$	9

$Trave(X')$	$Epid$	$Sick(X')$	g_2'
$false$	$false$	$true$	1
$false$	$true$	$true$	6
$true$	$false$	$true$	6
$true$	$true$	$true$	9

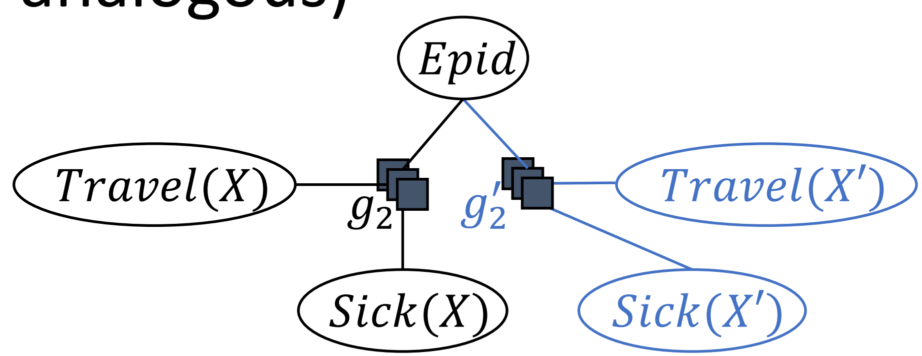
$Travel(X')$	$Epid$	g_2'
$false$	$false$	1
$false$	$true$	6
$true$	$false$	6
$true$	$true$	9

LJT: Enter Evidence

- At every parcluster that contains evidence variables
  - $g_E(Sick(X')), \mathcal{D}(X') = \{eve, alice\}$
  - Parclusters C_2 and C_3



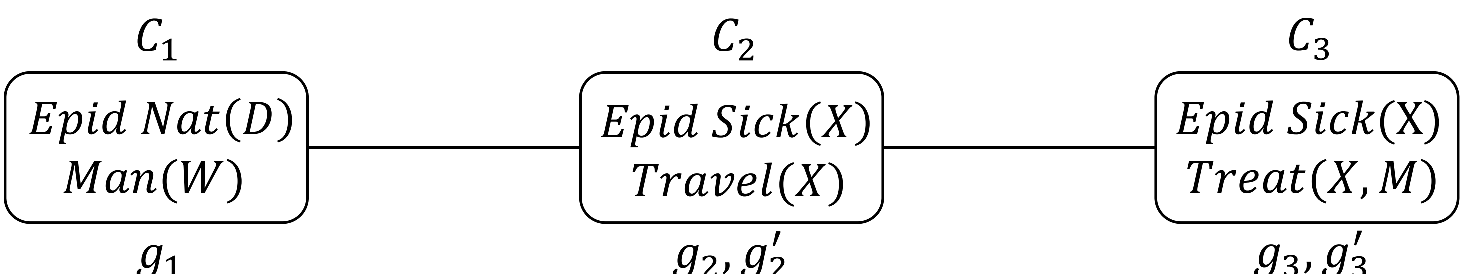
- Enter evidence at C_2 (C_3 analogous)
  - Split local model
    - $dom(X) = \{bob, \dots\}$
  - Absorb evidence in g_2'



Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017)

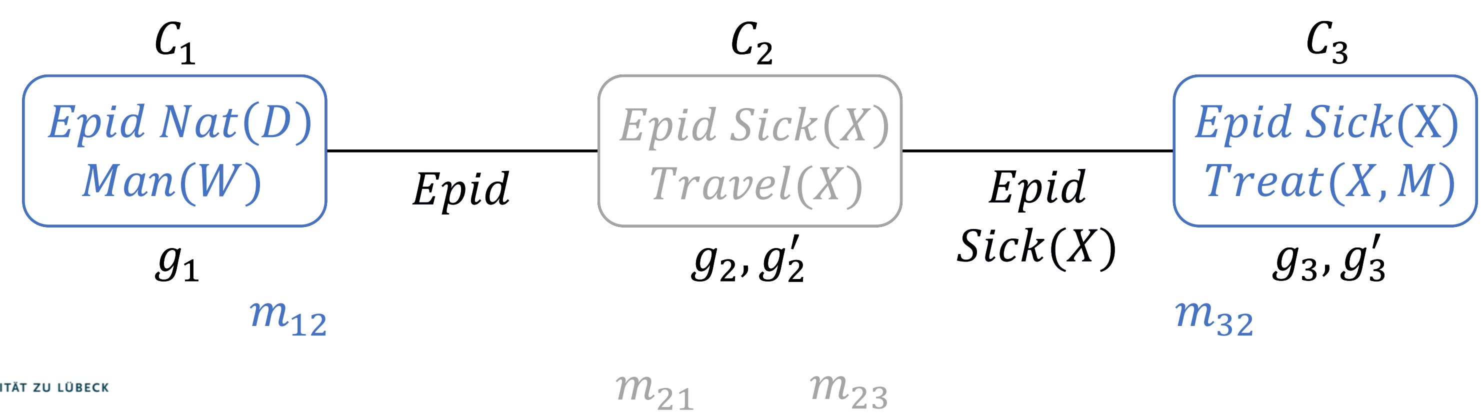
- Input
  - Model G
  - Evidence $\boldsymbol{E}$
  - Queries $\boldsymbol{Q}$
- Algorithm
  1. Build FO jtree J for G
  2. Enter evidence $\boldsymbol{E}$ into J



  3. Pass messages in J

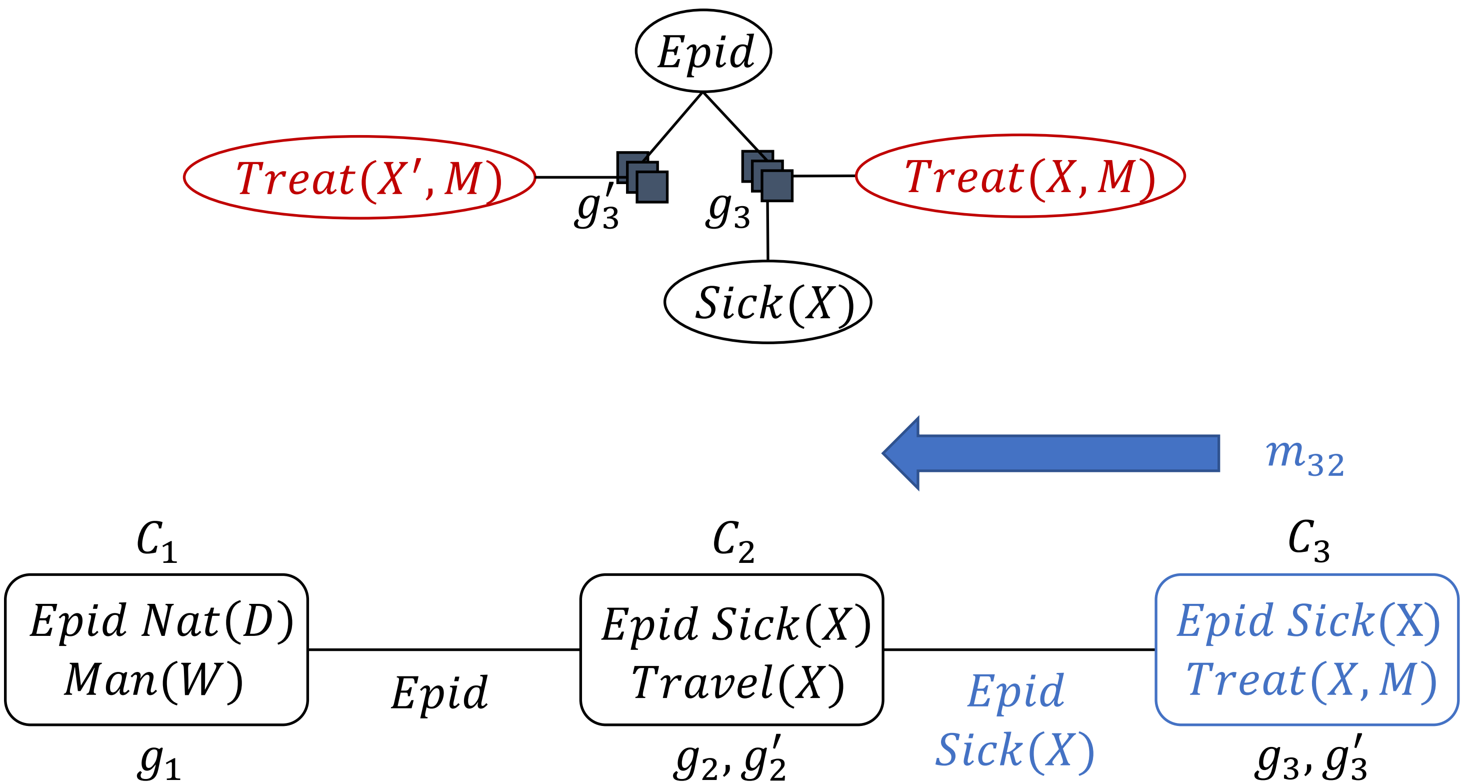
LJT: Pass Messages

- Separators
- Messages
  - Inbound
    - m_{12} from C_1 to C_2 over $Epid$
    - m_{32} from C_3 to C_2 over $Epid, Sick(X)$
  - Outbound
    - m_{21} from C_1 to C_2 over $Epid$
    - m_{23} from C_3 to C_2 over $Epid, Sick(X)$



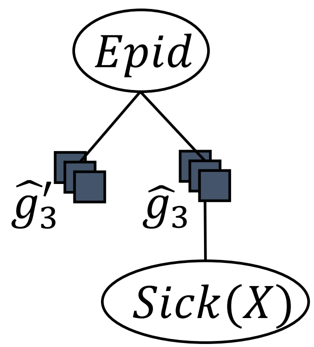
LJT: Example Message Inbound

- m_{32} from C_3 to C_2
  - Eliminate $Treat(X,P), Treat(X',P)$

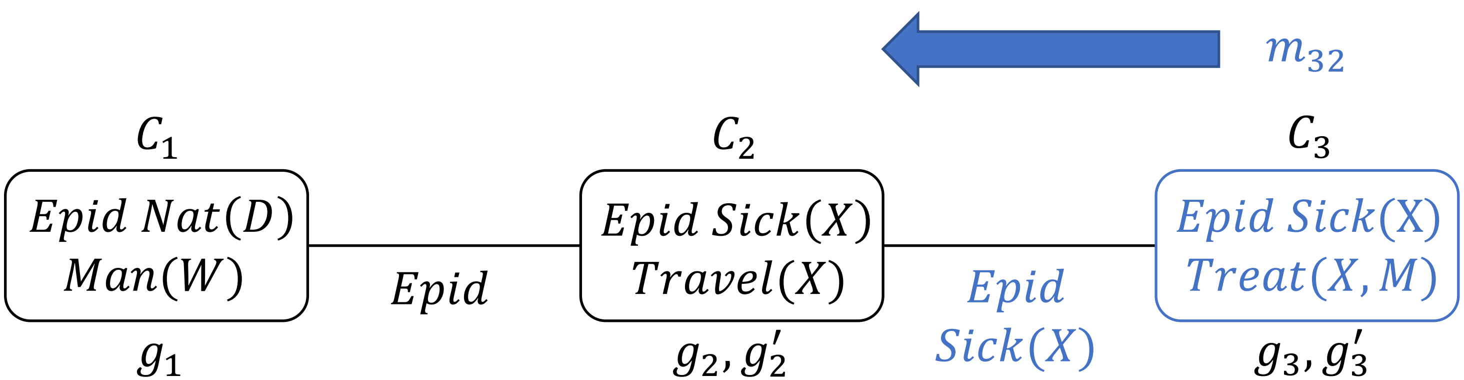


LJT: Example Message Inbound

- m_{32} from C_3 to C_2
  - Eliminate $Treat(X,P), Treat(X',P)$



$m_{32} = \{\hat{g}_3, \hat{g}_3'\}$



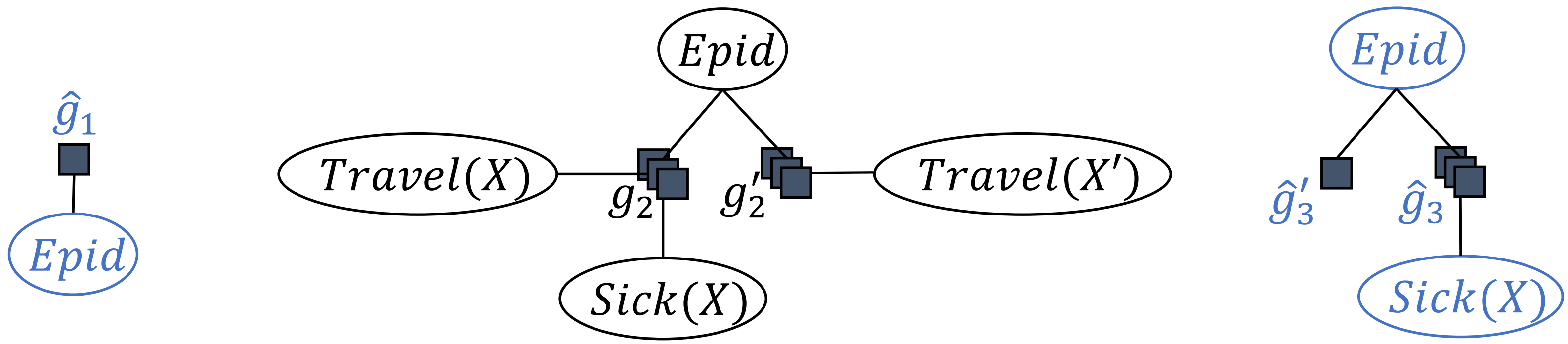
UNIVERSITÄT ZU LÜBECK

LJT: Messages at C_2

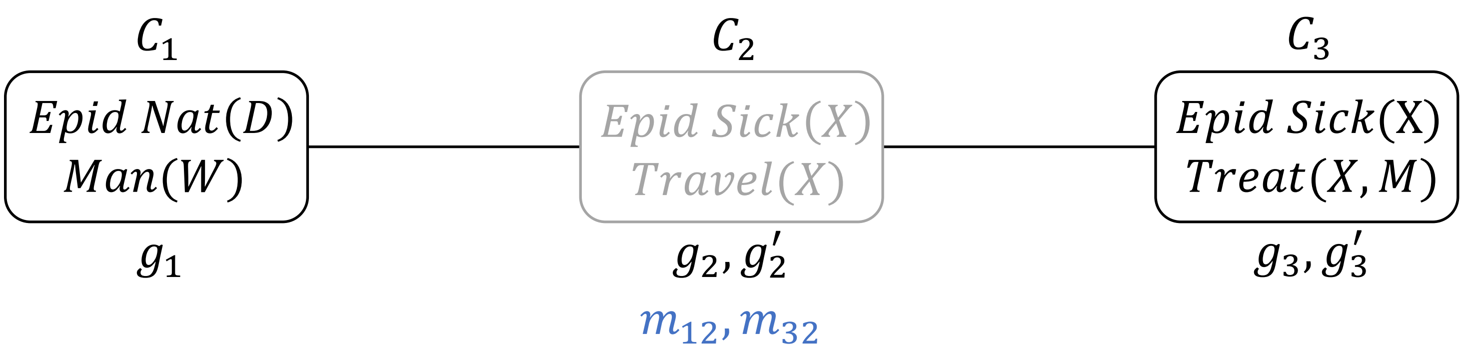
- After m_{12} and m_{32} arrived

$m_{12} = \{\hat{g}_1\}$

$m_{32} = \{\hat{g}_3, \hat{g}_3'\}$

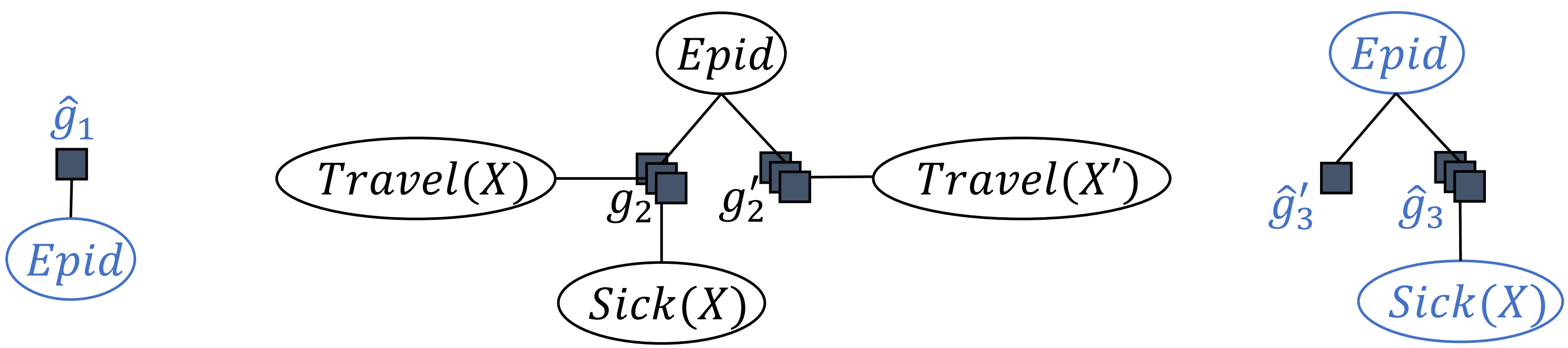


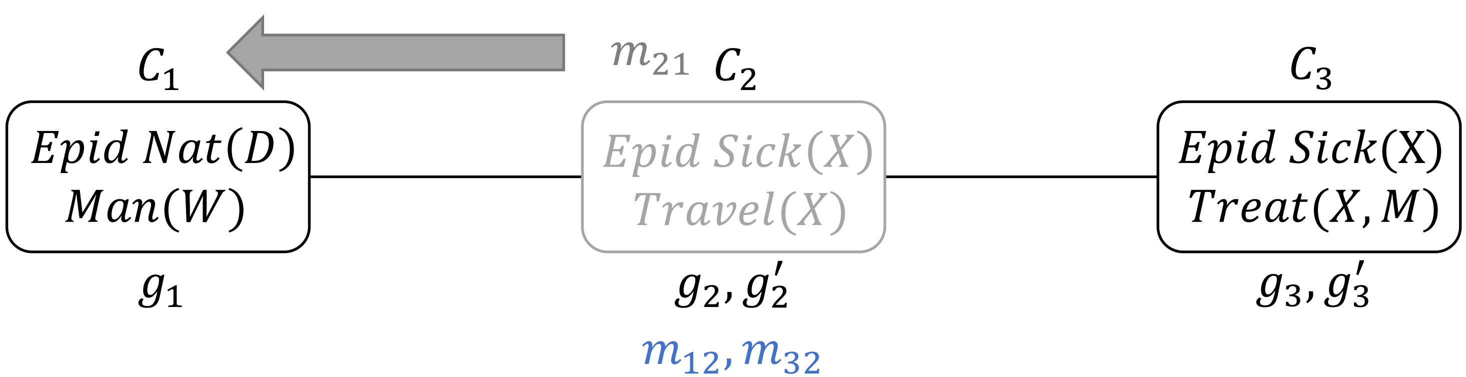
C_2 is now independent of C_1 and C_3



LJT: Example Message Outbound

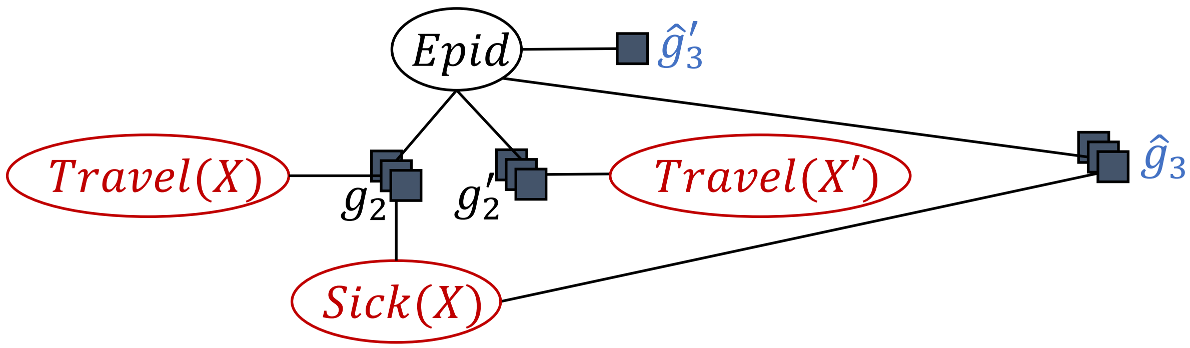
- m_{21} from C_2 to C_1
  - Eliminate $Sick(X), Travel(X), Travel(X')$ from g_2, g_2', m_{32}

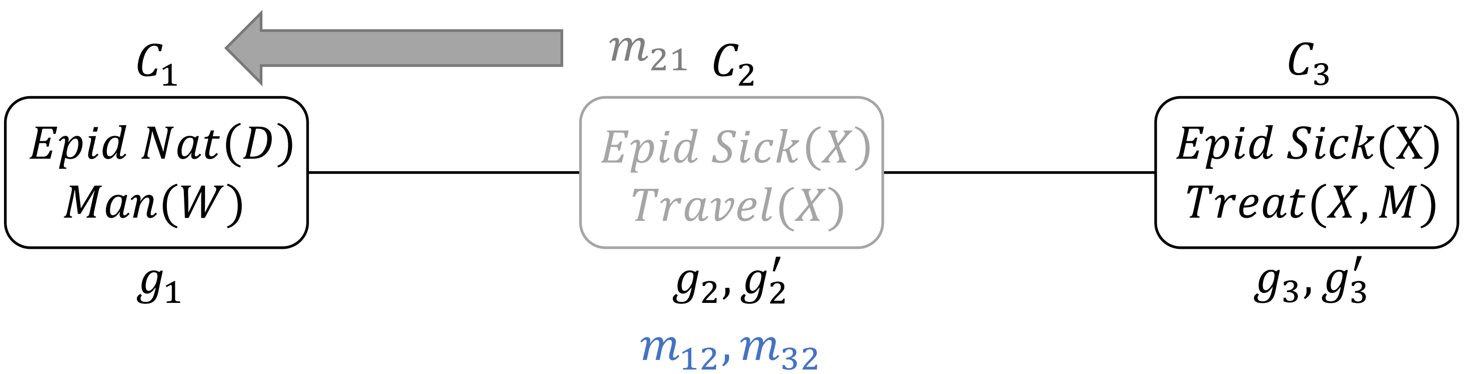




LJT: Example Message Outbound

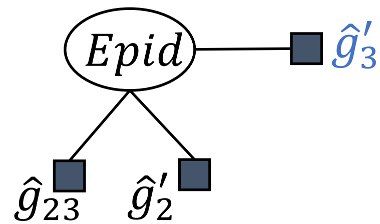
- m_{21} from C_2 to C_1
  - Eliminate $Sick(X), Travel(X), Travel(X')$ from g_2, g_2', m_{32}

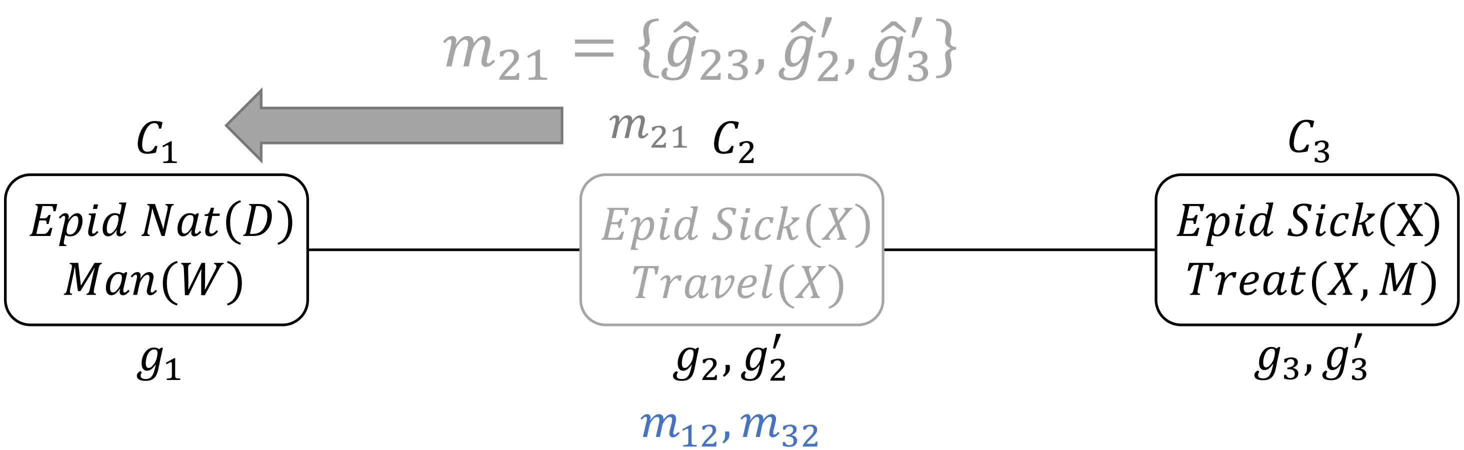




LJT: Example Message Outbound

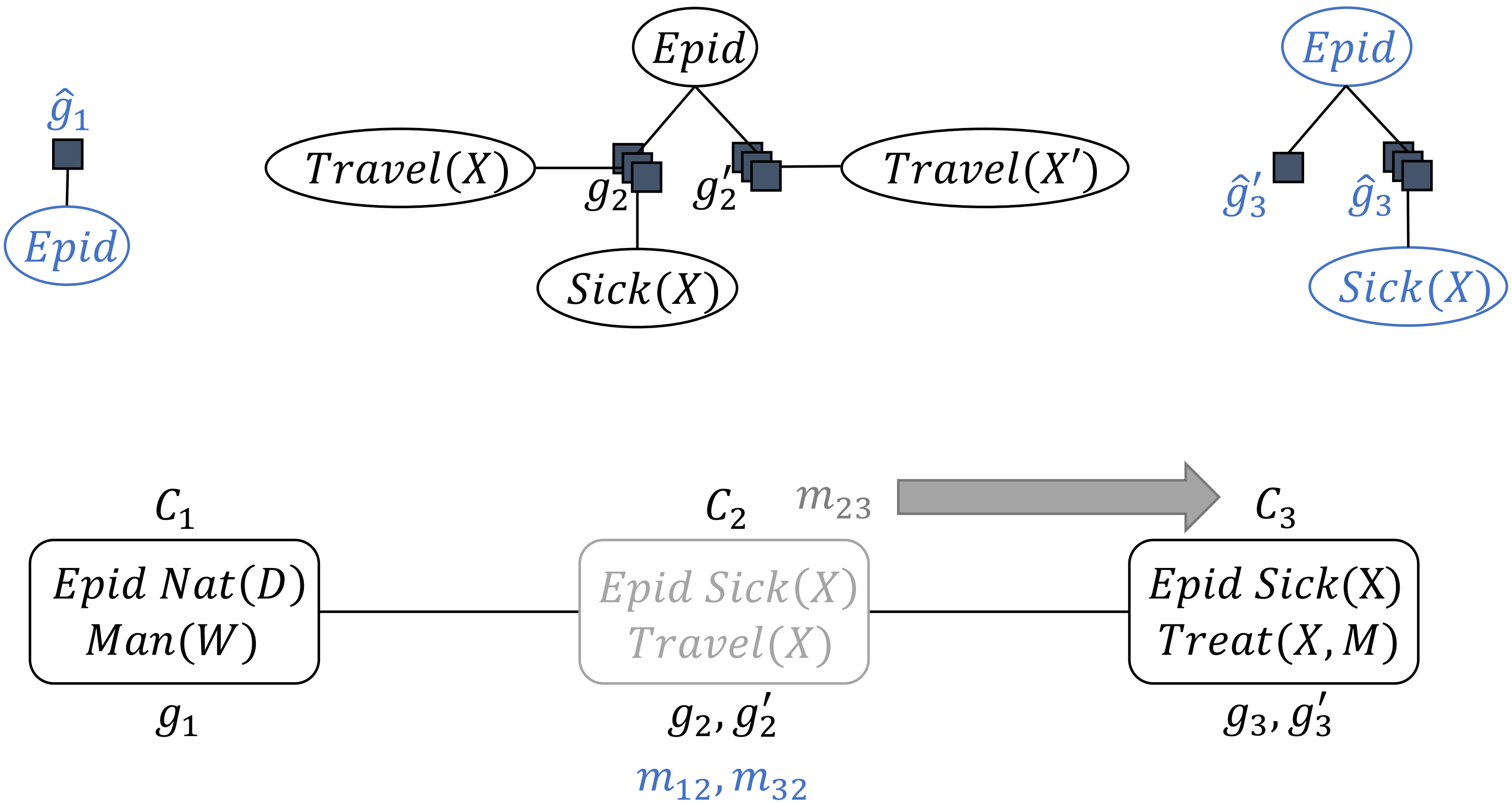
- m_{21} from C_2 to C_1
  - Eliminate $Sick(X), Travel(X), Travel(X')$ from g_2, g_2', m_{32}





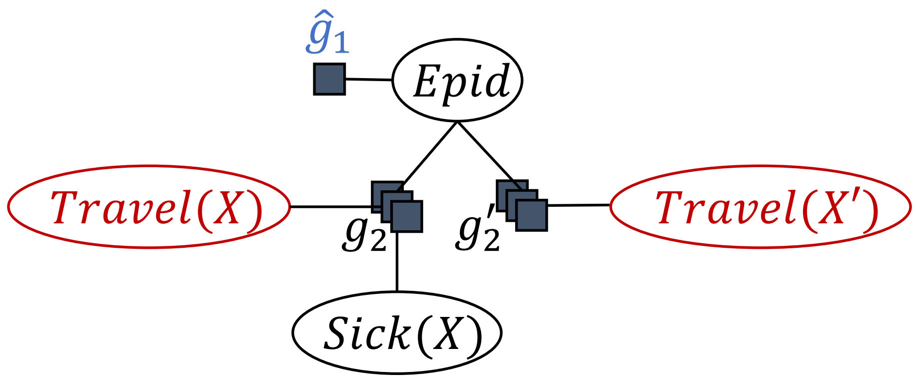
LJT: Example Message Outbound

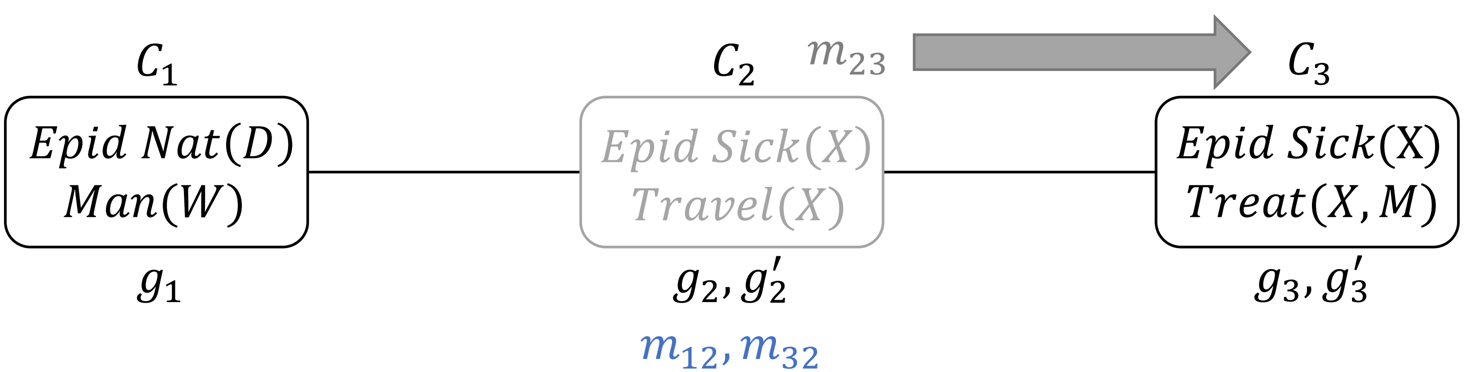
- m_{23} from C_2 to C_3
  - Eliminate $Travel(X), Travel(X')$ from g_2, g_2', m_{12}



LJT: Example Message Outbound

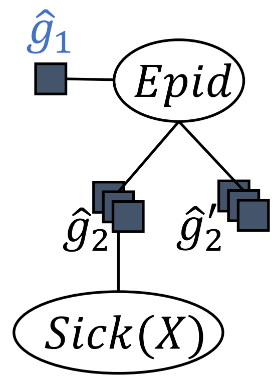
- m_{23} from C_2 to C_3
  - Eliminate $Travel(X), Travel(X')$ from g_2, g_2', m_{12}



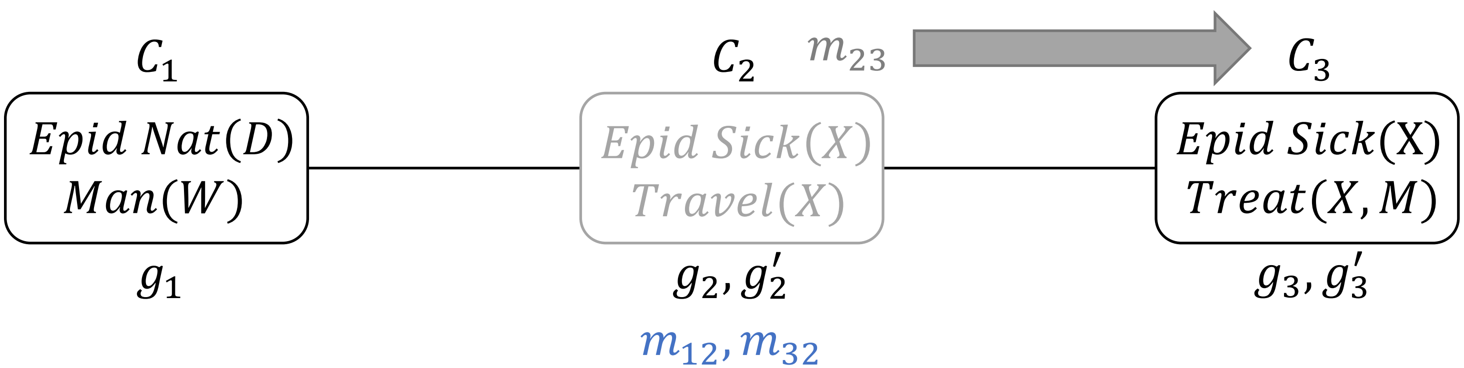


LJT: Example Message Outbound

- m_{23} from C_2 to C_3
  - Eliminate $Travel(X), Travel(X')$ from g_2, g_2', m_{12}



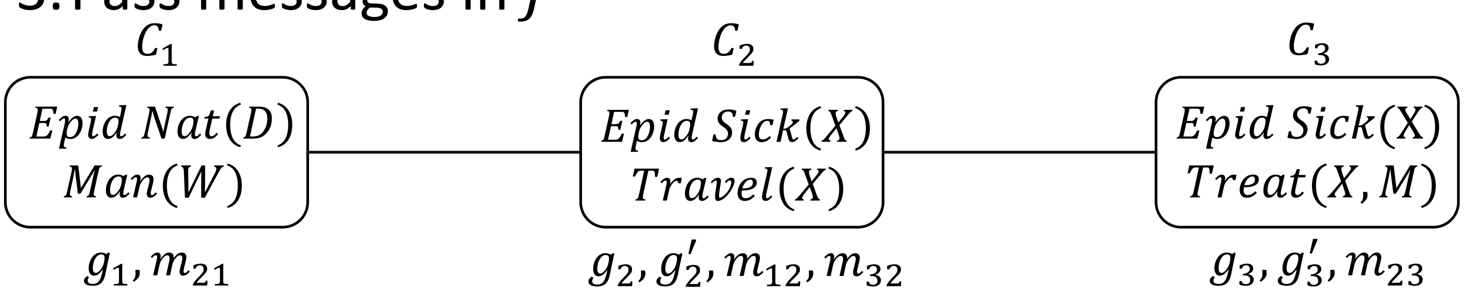
$$m_{23} = \{\hat{g}_1, \hat{g}_2, \hat{g}_2'\}$$



Lifted Junction Tree Algorithm: LJT

Braun and Möller (2017)

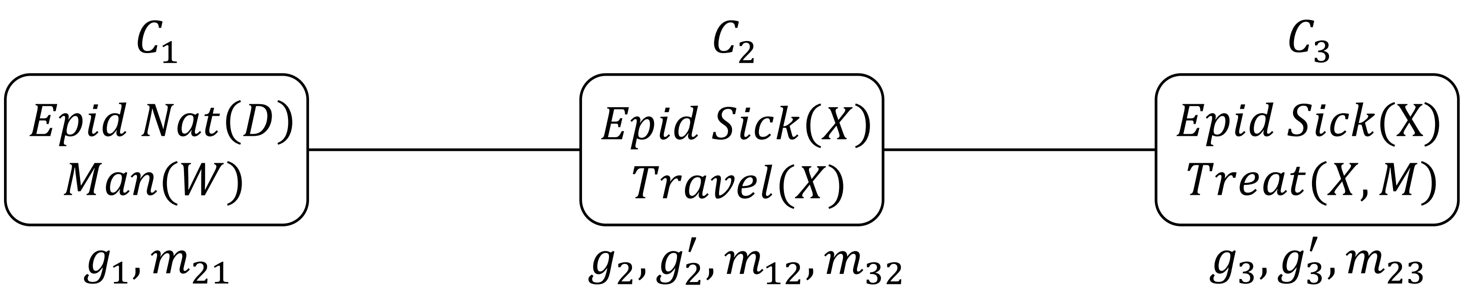
- Input
  - Model G
  - Evidence $\boldsymbol{E}$
  - Queries $\boldsymbol{Q}$
- Algorithm
  1. Build FO jtree J for G
  2. Enter evidence $\boldsymbol{E}$ into J
  3. Pass messages in J



  4. Answer queries $\boldsymbol{Q}$

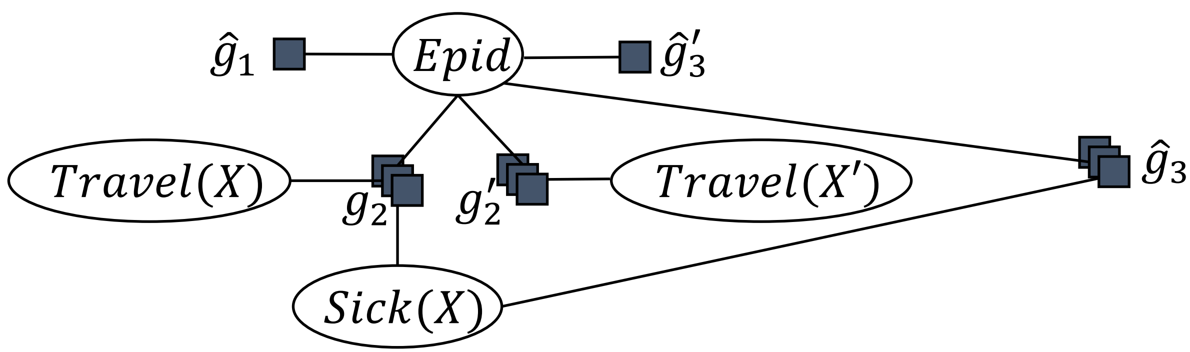
LJT: Answer Queries

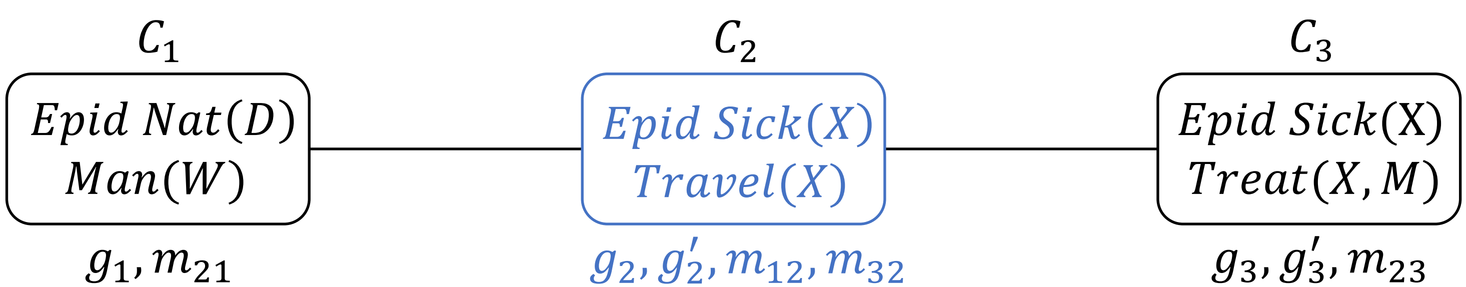
- Queries $\boldsymbol{Q} = \{Travel(eve), Epid\}$
- For each query Q
  - Find parcluster that contains Q
  - Extract submodel of local model and messages
  - Use LVE to answer Q



LJT: Answer Queries

- $Q_1 = Travel(eve)$
  - Find parcluster: C_2
  - Extract submodel: $G' = \{g_2, g_2', m_{12}, m_{32}\}$
  - Answer $Travel(eve)$ with LVE

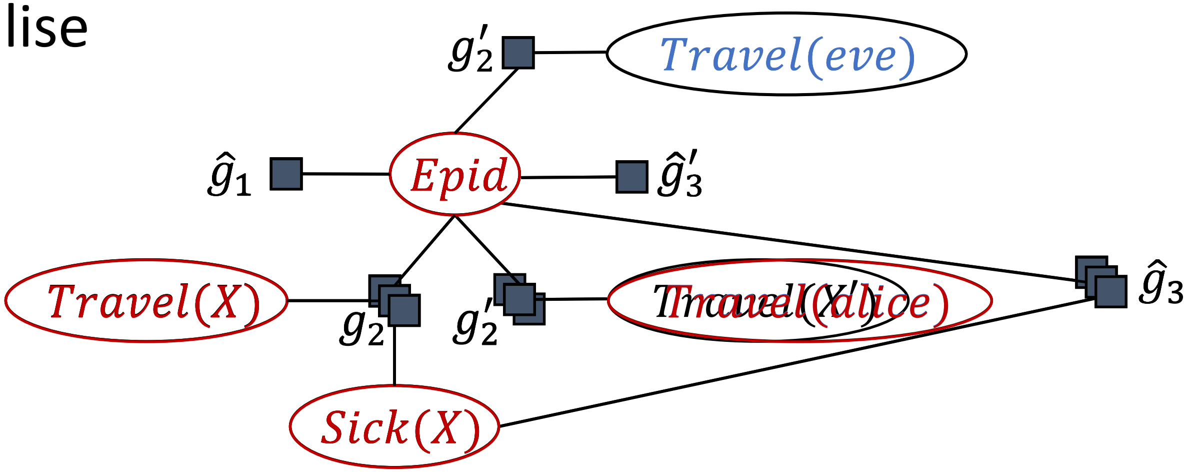




UNIVERSITÄT ZU LÜBECK

LJT: Answer Queries

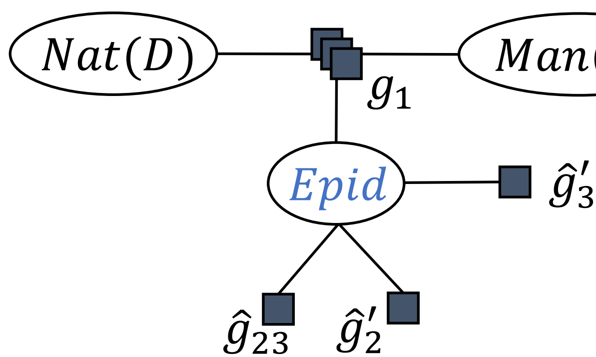
- Answer $Travel(eve)$ with LVE
  - Split model
  - Eliminate non-query variables
  - Normalise

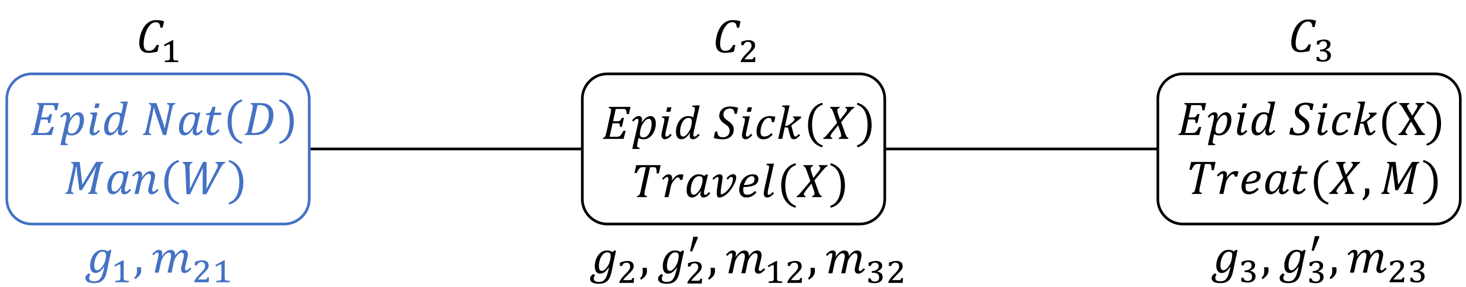


$$\mathcal{D}(X') = \{alice, eve\}$$
$$\mathcal{D}(X) = \{bob, \dots\}$$

LJT: Answer Queries

- $Q_2 = Epid$
  - Find parcluster: C_1 (any of the three parclusters)
  - Extract submodel: $G' = \{g_1, m_{21}\}$
  - Answer $Epid$ with LVE

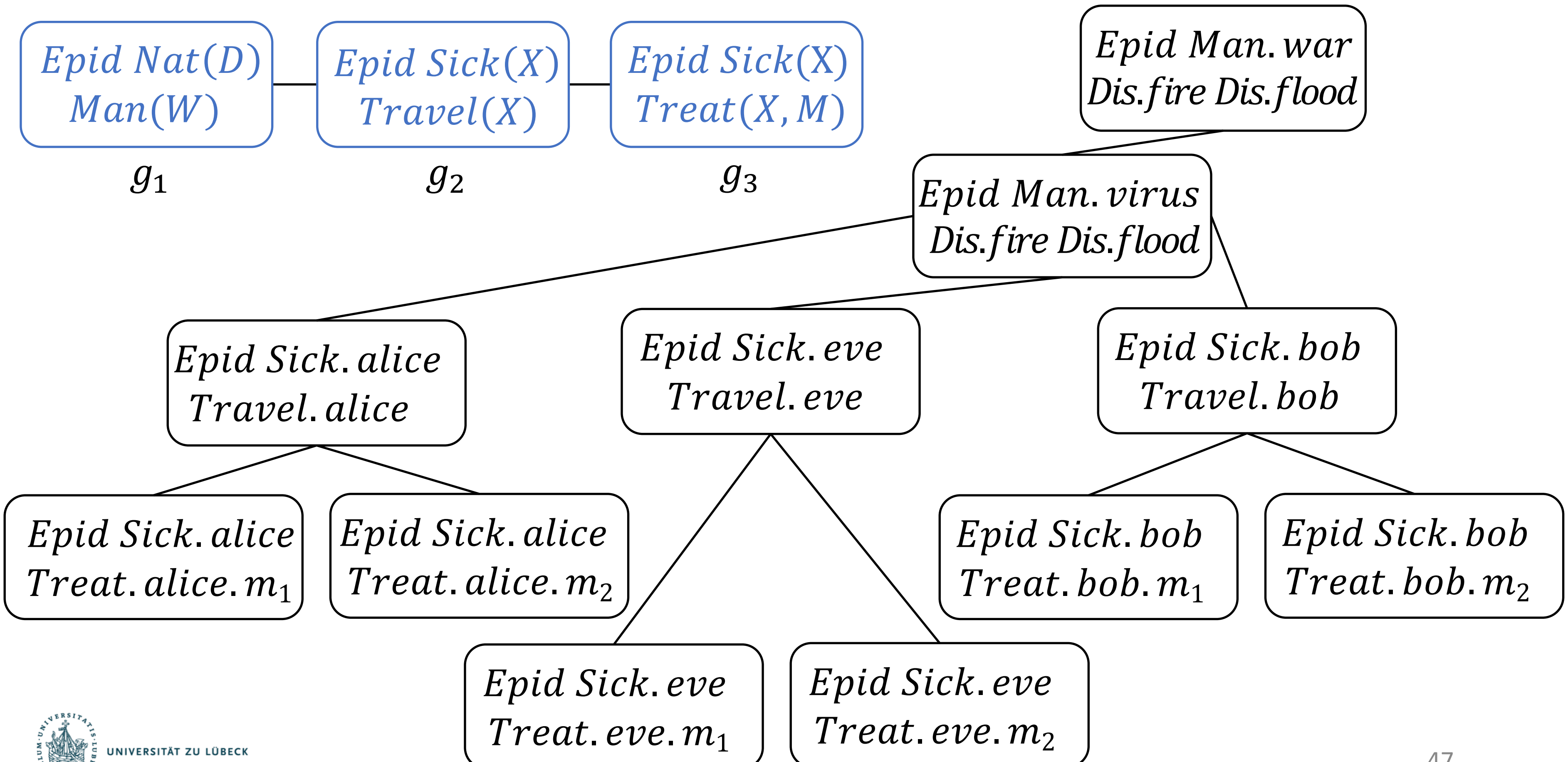




UNIVERSITÄT ZU LÜBECK

Lifting for Efficiency

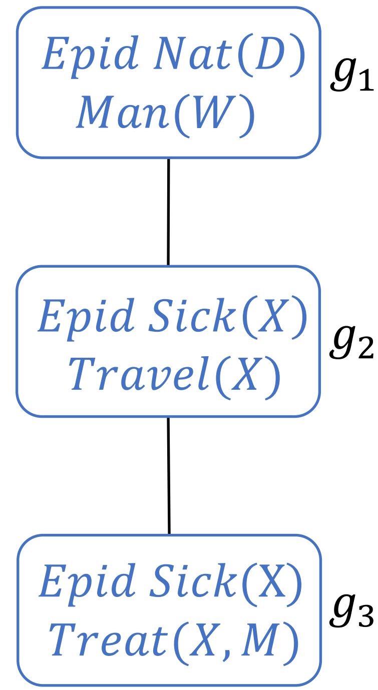
- Runtime efficiency: LVE in calculations
- In addition: space efficiency (nodes, messages)



Soundness & Completeness

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

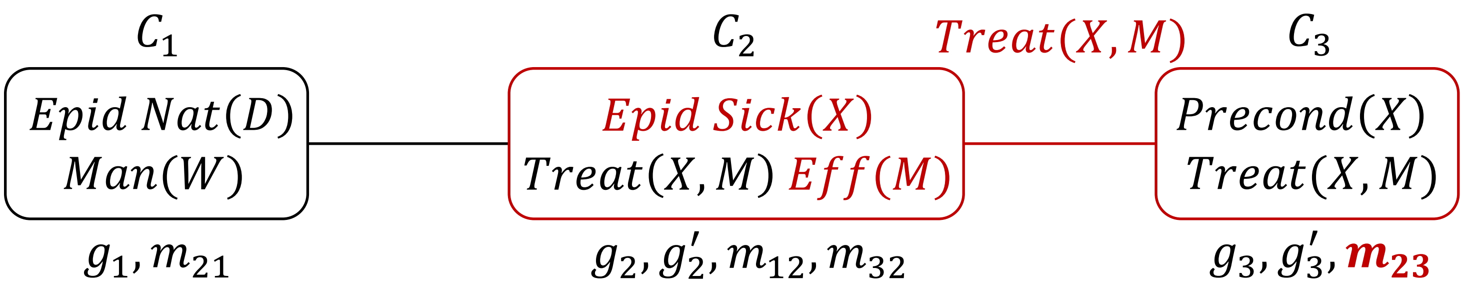
- Soundness
  - Local computations on nodes correct if
    - Valid junction tree (w.r.t. properties)
    - Combination & marginalisation
      (in form of multiplication & summing out)
  - Local computations for messages and queries
- Completeness
  - No groundings in any case
  - Two logical variables per parfactor
  - One logical variable per PRV (arbitrarily many logical variables per parfactor)
  - Holds for many lifted algorithms



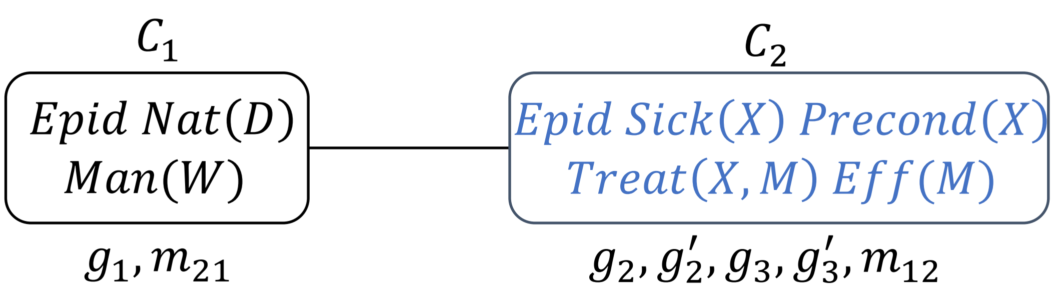
Is it that easy?

Braun and Möller (2017)

- Direct translation of propositional junction tree algorithm to lifted case yields groundings
  - Reason in precondition of lifted summing out: PRV to eliminate has to contain all logical variables of the parfactor



- Additional step: Fusion!



LJT: Analysis

- Static overhead
  - Construction
  - Evidence entering
  - Message passing
  - To avoid groundings, parclusters may need to be fused
- Payoff during QA
  - Multiple queries
  - Without groundings
  - Complexity of LVE for one query
    = Complexity of message pass in LJT

Queries all under the same evidence

$$\boldsymbol{E} = \{Sick(eve) = true,$$
$$Sick(alice) = true\}$$

Extending LVE and LJT

Braun and Möller (2017a, 2018, 2018a, 2018b), Gehrke et al. (2019)

- Adaptive inference (incremental changes)
  - Evidence, model structure, parfactors $\longrightarrow$ Adaptive steps of LJT
- Conjunctive queries
  - $P\big(Epid, Travel(eve)\big)$
- Isomorphic query terms (parameterised queries)
  - $P\big(Sick(eve), Sick(alice), Sick(bob)\big) \triangleq P\big(Sick(X)\big)$
- Most probable assignment (MPE, MAP)
  - New argmax operators
- Uncertain evidence
  - $Sick(eve) = true$ with probability of 0.9

UNIVERSITÄT ZU LÜBECK

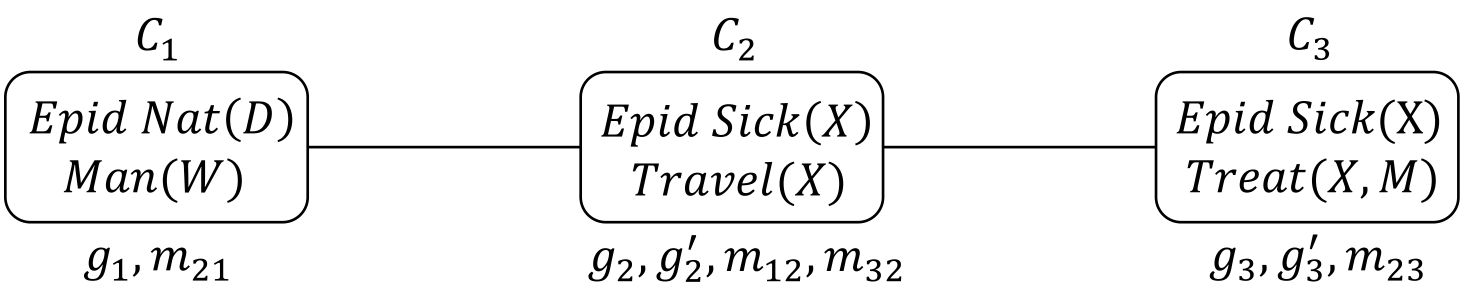
Does it have to be LVE in LJT?

Braun and Möller (2018c)

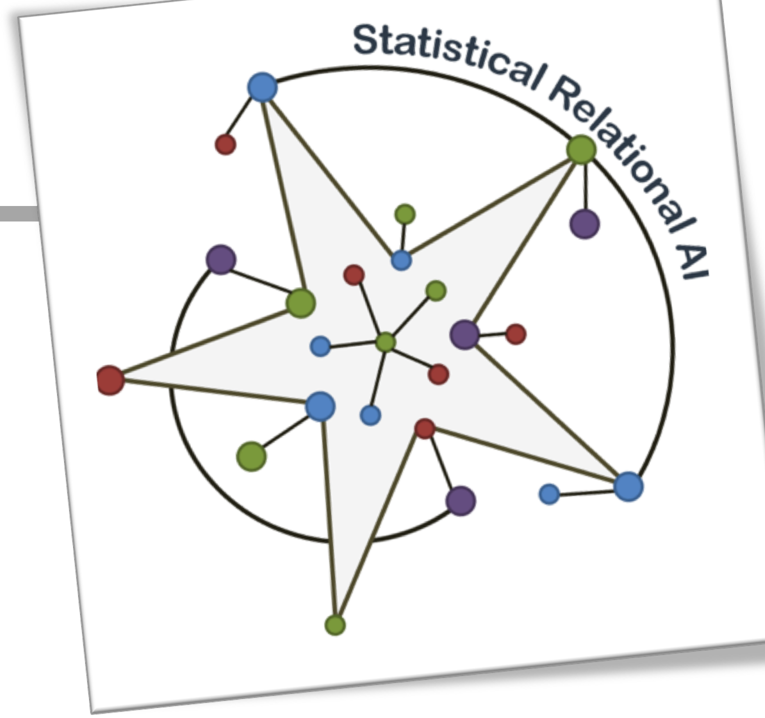
LJT with LVE &

First-order Knowledge Compilation (FOKC)
to solve a WFOMC problem

- LVE for evidence entering and message passing
- FOKC for query answering



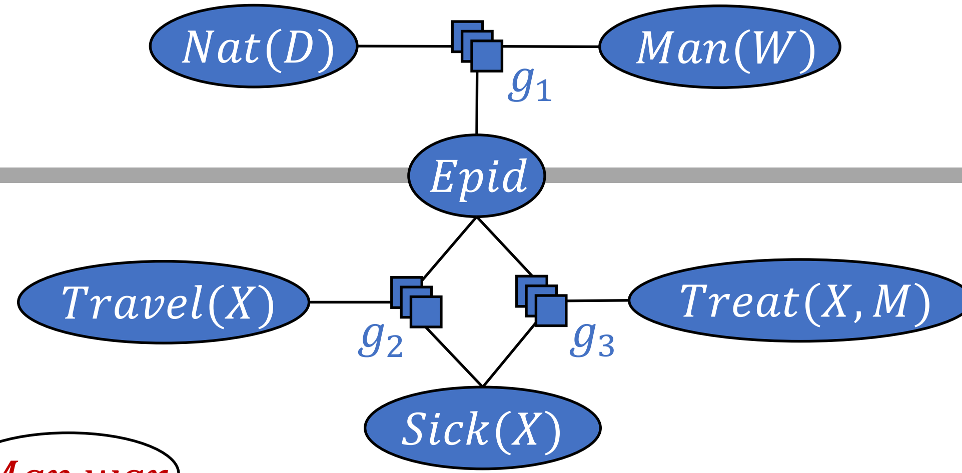
- Other lifted algorithms to replace LVE in LJT…



But...

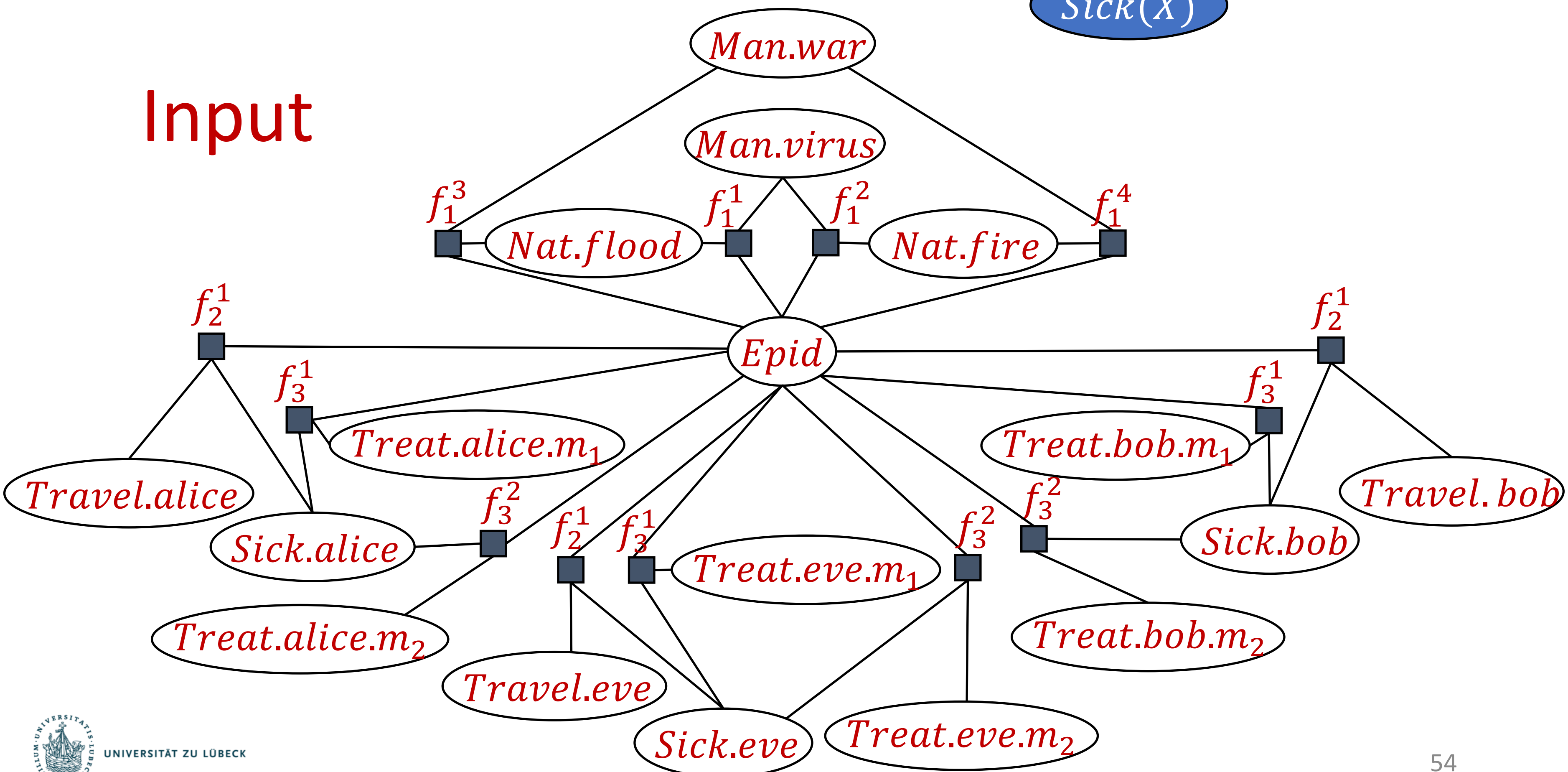
What if there is only a propositional model?

Compression



Goal

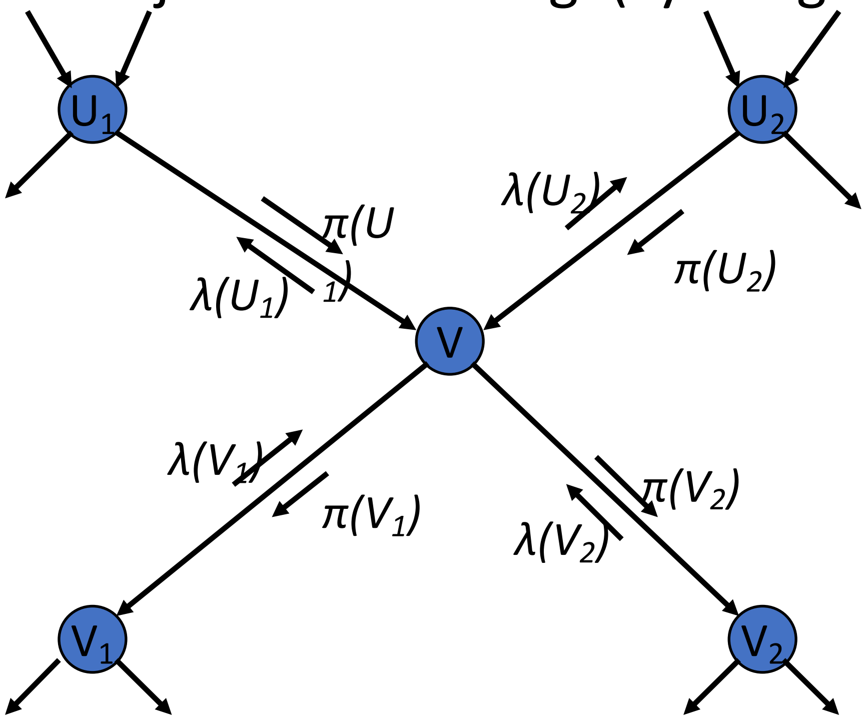
Input



A Bit of History…

Pearl (1982)

- Pearl's Belief propagation
  - Messages on Bayes net
  - Exact for polytrees (no cycles in undirected graph!)
  - Precursor of junction tree alg. (cycles go into clusters)



Loopy Belief Propagation

Singla and Domingos (2008), Kersting et al. (2009), Ahmadi et al. (2013)

- Pass messages on graph
  - If no cycles: exact
  - Else: approximate

- Lifted (loopy) belief propagation
  - Exploit computational symmetries
  - Compress graph whenever nodes would send identical messages
  - Send messages on compressed graph

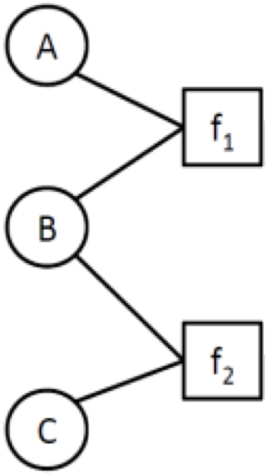
→ Colour passing algorithm for compression

Compression: Pass the colours around*

Singla and Domingos (2008), Kersting et al. (2009), Ahmadi et al. (2013)

- **Colour nodes according to the evidence you have**
  - No evidence, say **red**
  - State „one", say **brown**
  - State „two", say **orange**
  - ...

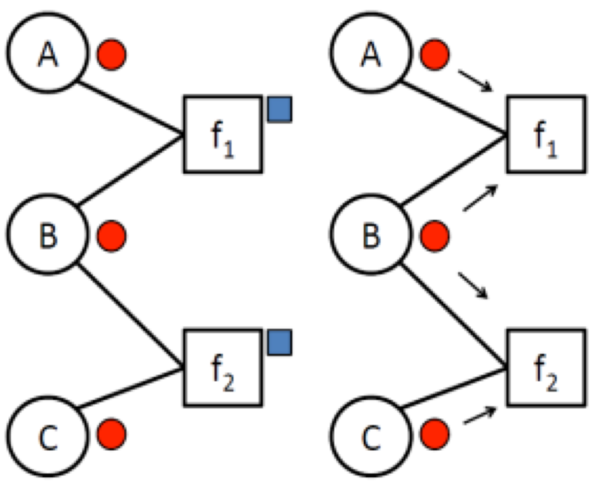
- **Colour factors distinctively according to their equivalences**
  For instance, assuming f_1 and f_2 to be identical and B appears at the second position within both, say **blue**



*can also be done at the „lifted", i.e., relational level

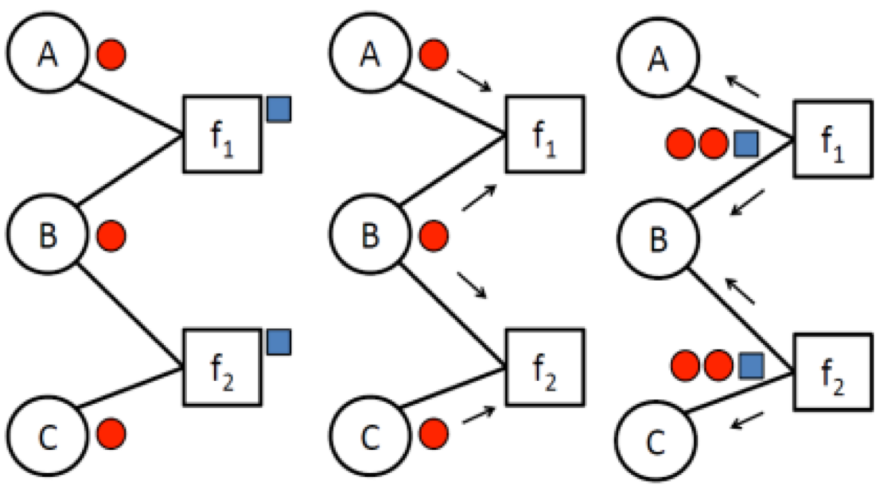
Compression: Pass the colours around

1. Each factor collects the colours of its neighbouring nodes



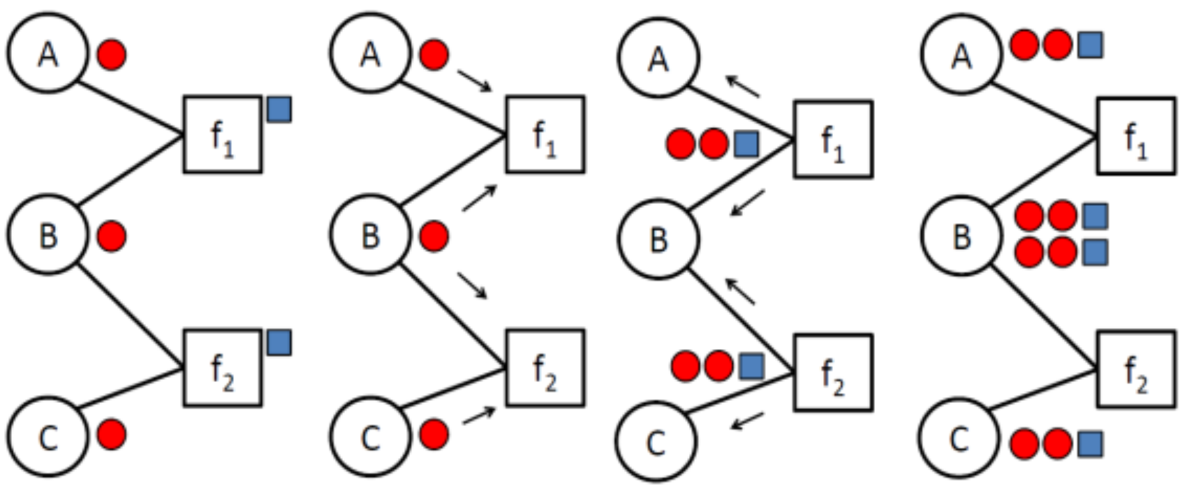
Compression: Pass the colours around

1. Each factor collects the colours of its neighbouring nodes
2. Each factor „signs“ its colour signature with its own colour



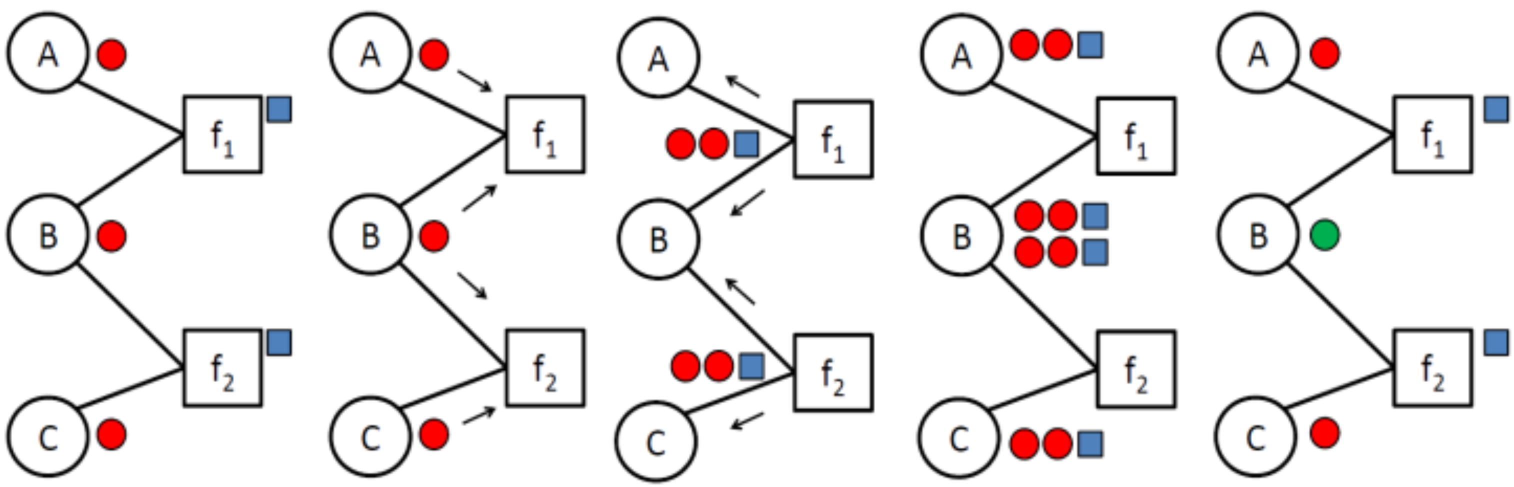
Compression: Pass the colours around

1. Each factor collects the colours of its neighbouring nodes
2. Each factor „signs“ its colour signature with its own colour
3. Each node collects the signatures of its neighbouring factors



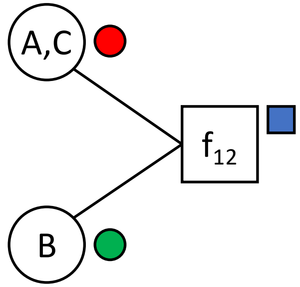
Compression: Pass the colours around

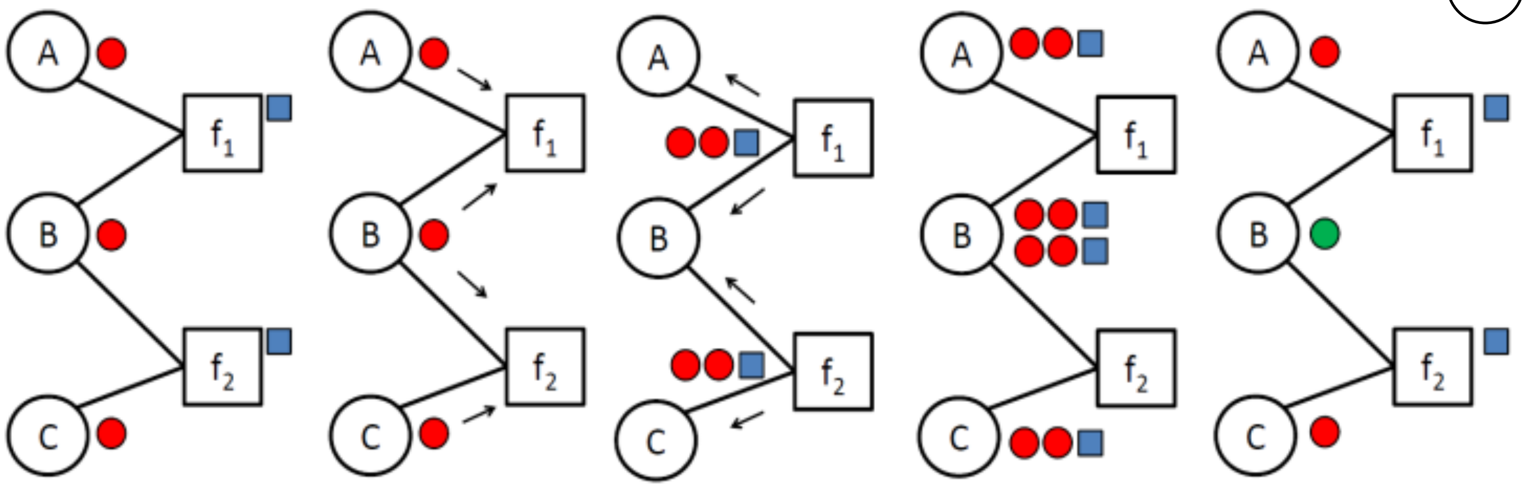
1. Each factor collects the colours of its neighbouring nodes
2. Each factor „signs“ its colour signature with its own colour
3. Each node collects the signatures of its neighbouring factors
4. Nodes are recoloured according to the collected signatures



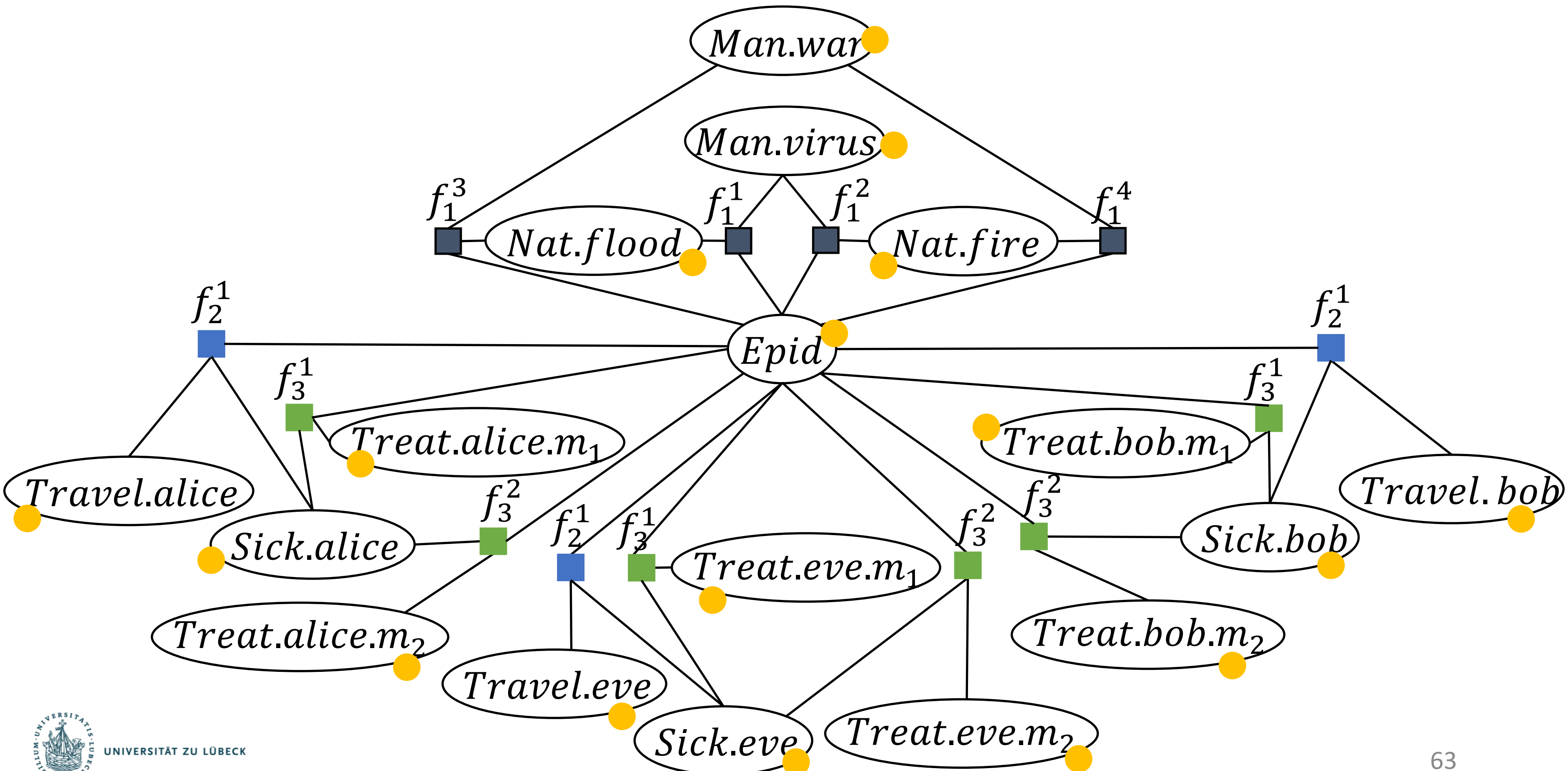
Compression: Pass the colours around

1. Each factor collects the colours of its neighbouring nodes
2. Each factor „signs“ its colour signature with its own colour
3. Each node collects the signatures of its neighbouring factors
4. Nodes are recoloured according to the collected signatures
5. If no new colour is created stop, otherwise go back to 1

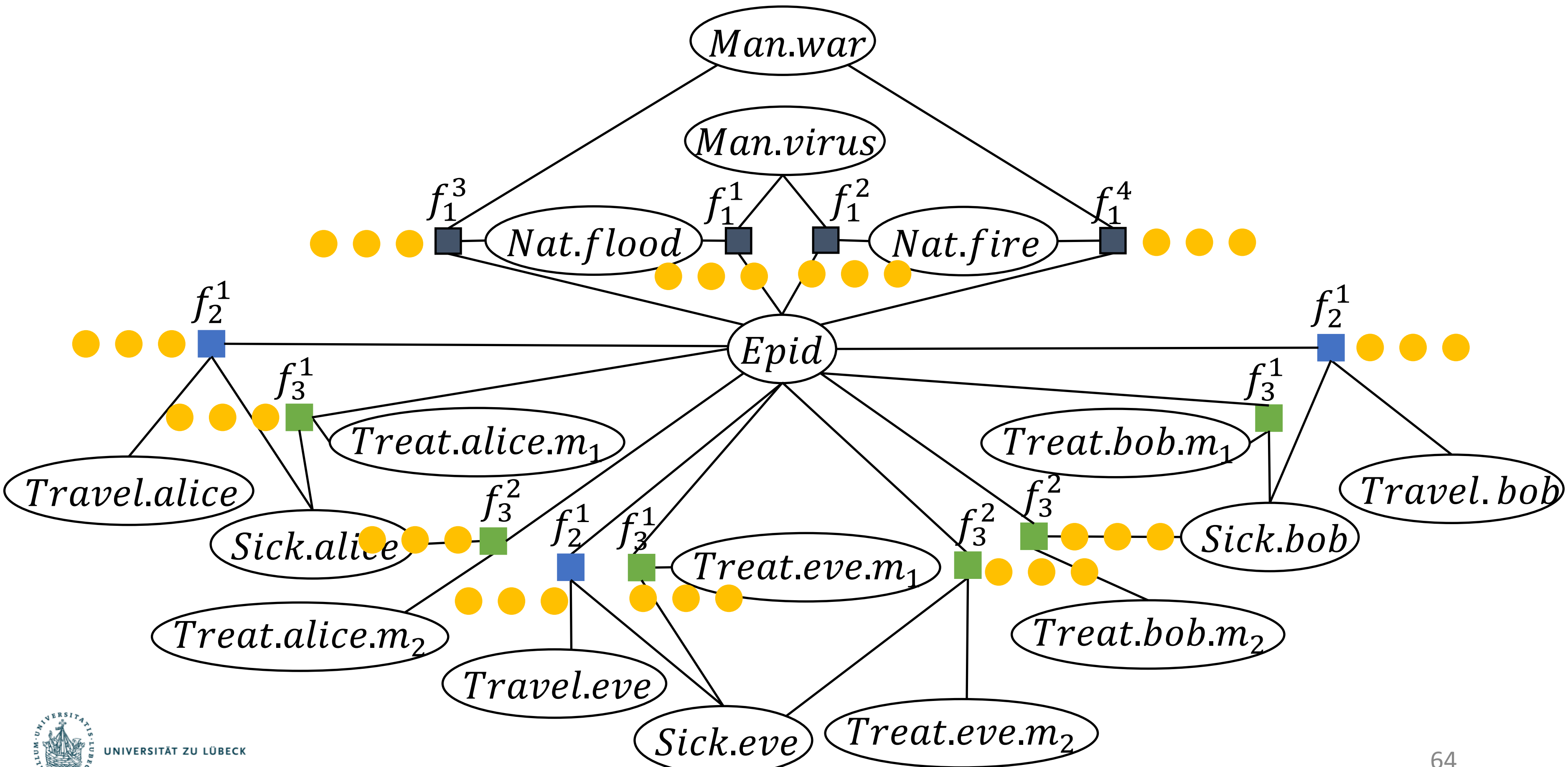




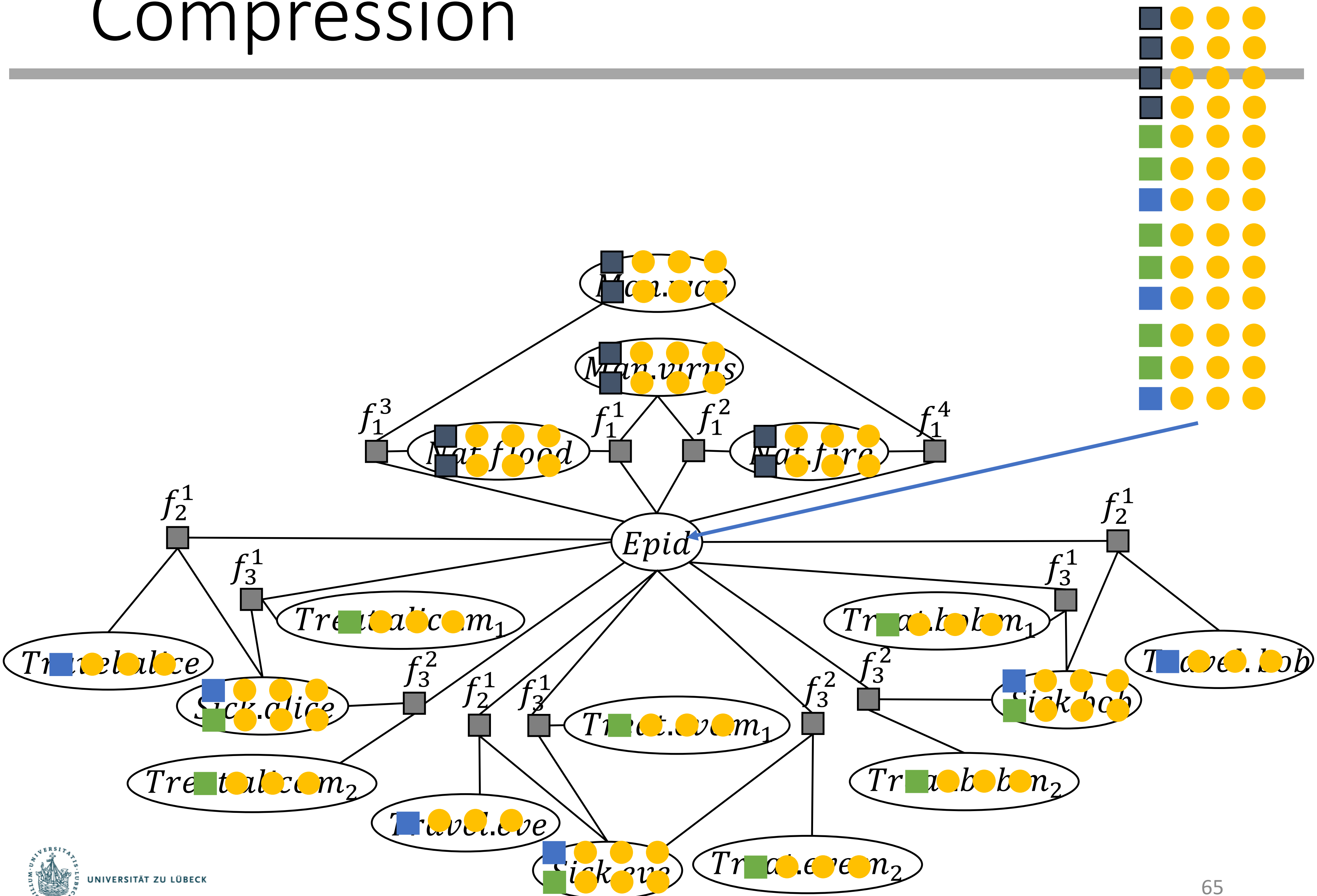
Compression



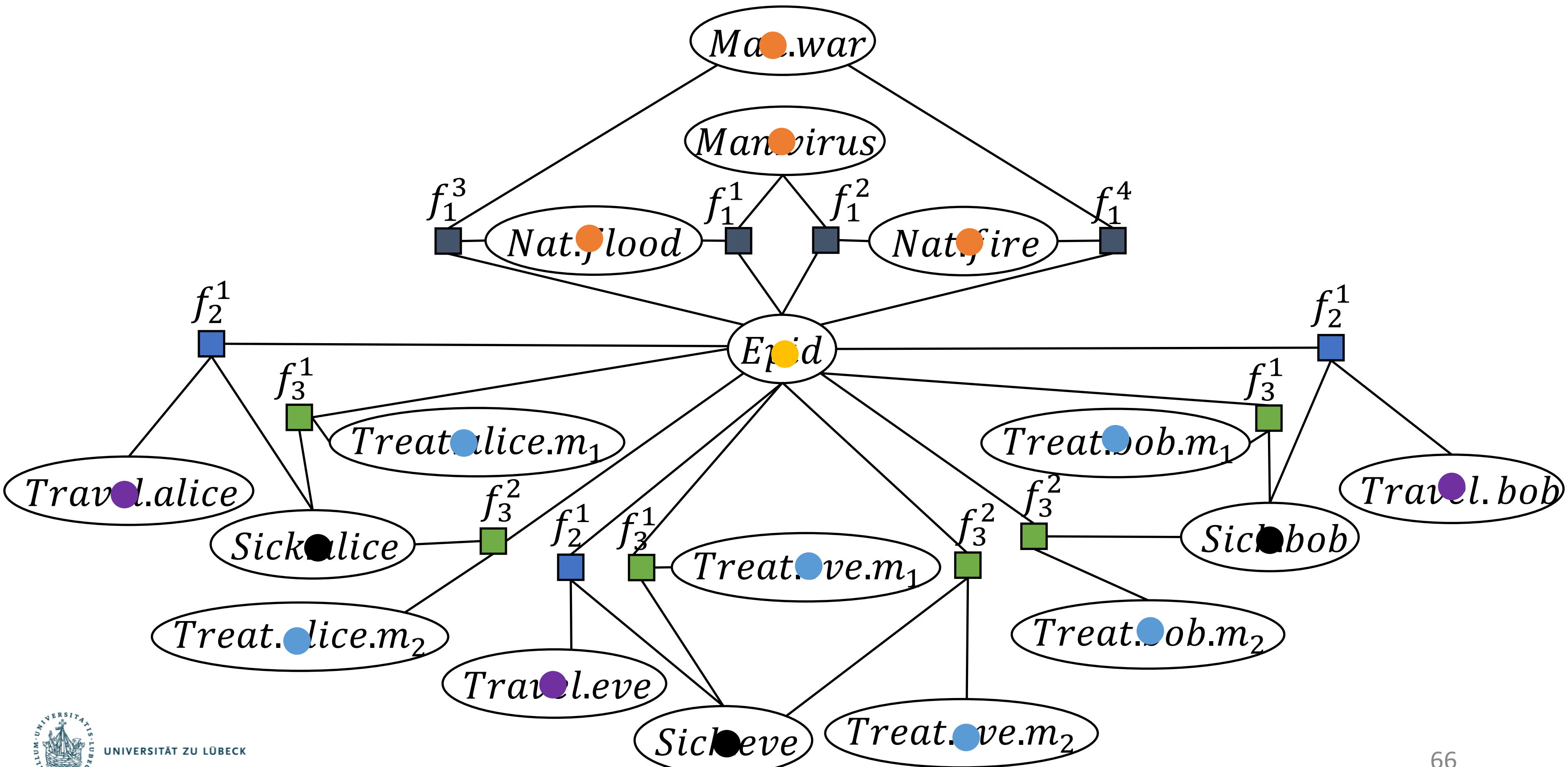
Compression



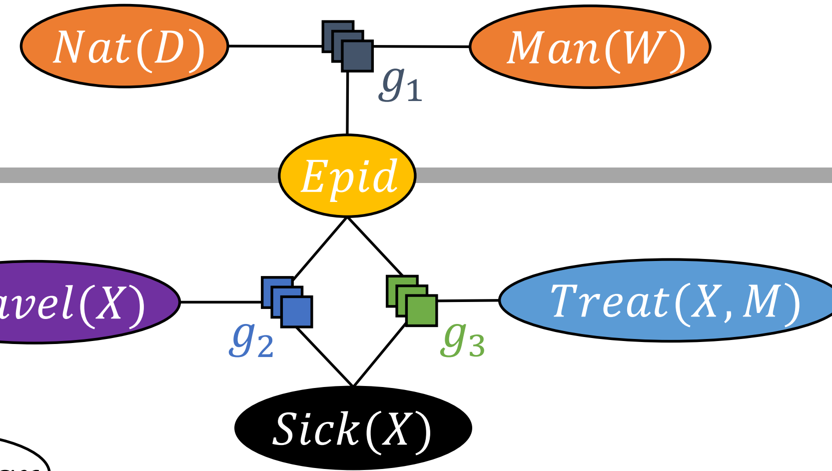
Compression

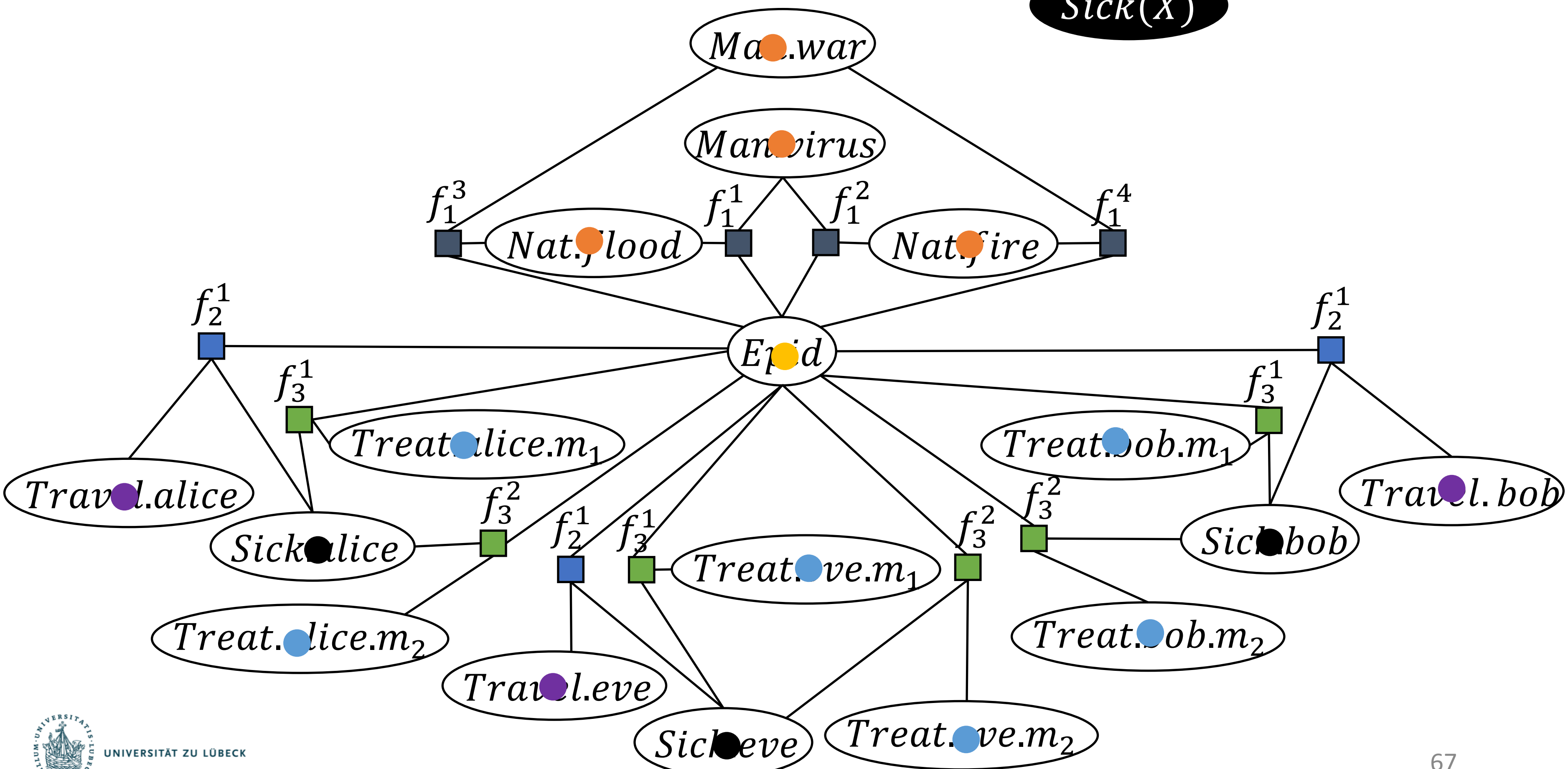


Compression



Compression





Wrap-up Exact Lifted Inference

- Algorithms for exact query answering on PRMs
  - LVE for single inference
    - Using lifting for efficiency w.r.t. domain sizes
  - LJT for repeated inference
    - Using smaller models for efficiency over multiple queries
  - Extensions possible
- Colour passing for compressing propositional models

Next: Answering Continuous Queries in DPRMs

References

And own work

References

- Ahmadi et al. (2013)
  Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. In *Machine Learning*. 92(1):91-132, 2013.
- De Salvo Braz et al. (2005)
  Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted First-order Probabilistic Inference. *IJCAI-05 Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005.
- De Salvo Braz et al. (2006)
  Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. MPE and Partial Inversion in Lifted Probabilistic Variable Elimination. *AAAI-06 Proceedings of the 21st Conference on Artificial Intelligence*, 2006.
- Jensen et al. (1990)
  Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. Bayesian Updating in Recursive Graphical Models by Local Computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

References

- Jensen et al. (1990)
  Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. Bayesian Updating in Recursive Graphical Models by Local Computations. *Computational Statistics Quarterly*, 4:269–282, 1990.
- Kersting et al. (2009)
  Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting Belief Propagation. In *UAI-09 Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.
- 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.
- Milch et al. (2008)
  Brian Milch, Luke S. Zettelmoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted Probabilistic Inference with Counting Formulas. In *AAAI-08 Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.

References

- Pearl (1982)
  Judea Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *AAAI-82 Proceedings of the 2nd National Conference on Artificial Intelligence*, 1982.
- Poole (2003)
  David Poole. First-order Probabilistic Inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.
- Shafer and Shenoy (1989)
  Glenn R. Shafer and Prakash P. Shenoy. Probability Propagation. In *Annals of Mathematics and Artificial Intelligence*, 2(1):327–351, 1989.
- Shenoy and Shafer (1990)
  Prakash P. Shenoy and Glenn R. Shafer. Axioms for Probability and Belief-Function Propagation. In *Uncertainty in Artificial Intelligence 4*, 9:169–198, 1990.

References

- Singla and Domingos (2008)
  Parag Singla and Pedro Domingos. Lifted First-order Belief Propagation. In *AAAI-08 Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008.
- Taghipour et al. (2013)
  Nima Taghipour, Jesse Davis, and Hendrik Blockeel. A Generalized Counting for Lifted Variable Elimination. In *Proceedings of the International Conference on Inductive Logic Programming*, pages 107–122, 2013.
- Taghipour et al. (2013a)
  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. (2013b)
  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.

Work @ IFIS

- 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. 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 (2017a)
  Tanya Braun and Ralf Möller. Counting and Conjunctive Queries in the Lifted Junction Tree Algorithm. In *Postproceedings of the 5th International Workshop on Graph Structures for Knowledge Representation and Reasoning*, 2017.
- Braun and Möller (2018)
  Tanya Braun and Ralf Möller. Adaptive Inference on Probabilistic Relational Models. In *Proceedings of the 31st Australasian Joint Conference on Artificial Intelligence*, 2018.

Work @ IFIS

- Braun and Möller (2018a)
  Tanya Braun and Ralf Möller. Parameterised Queries and Lifted Query Answering. In *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.
- Braun and Möller (2018b)
  Tanya Braun and Ralf Möller. Lifted Most Probable Explanation. In *Proceedings of the International Conference on Conceptual Structures*, 2018.
- Braun and Möller (2018c)
  Tanya Braun and Ralf Möller. Fusing First-order Knowledge Compilation and the Lifted Junction Tree Algorithm. In *Proceedings of KI 2018: Advances in Artificial Intelligence*, 2018.
- Gehrke et al. (2019)
  Marcel Gehrke, Tanya Braun, and Ralf Möller. Uncertain Evidence in Probabilistic Relational Models. In *Proceedings of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, 2019, to appear.