UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Lifted Junction Tree Algorithm

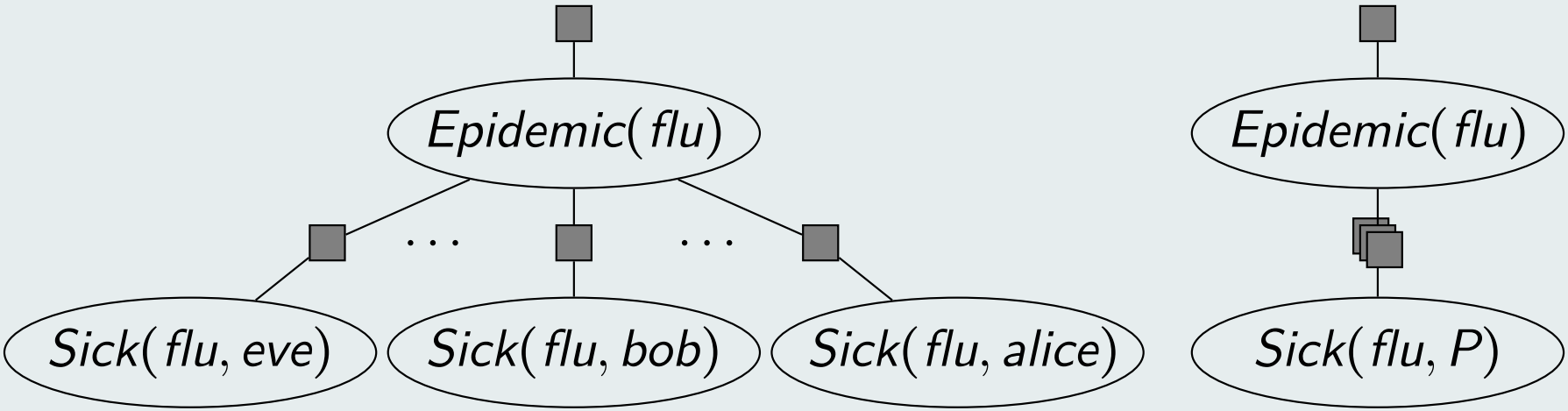
An Overview

Tanya Braun, Ralf Möller

Institute of Information Systems
Universität zu Lübeck

September 29, 2016

(Parameterised) Model



Ground factors

$\phi(Epidemic(flu))$

$\phi(Sick(flu, eve), Epidemic(flu))$

...

Parfactor

$\phi(Epidemic(flu))$

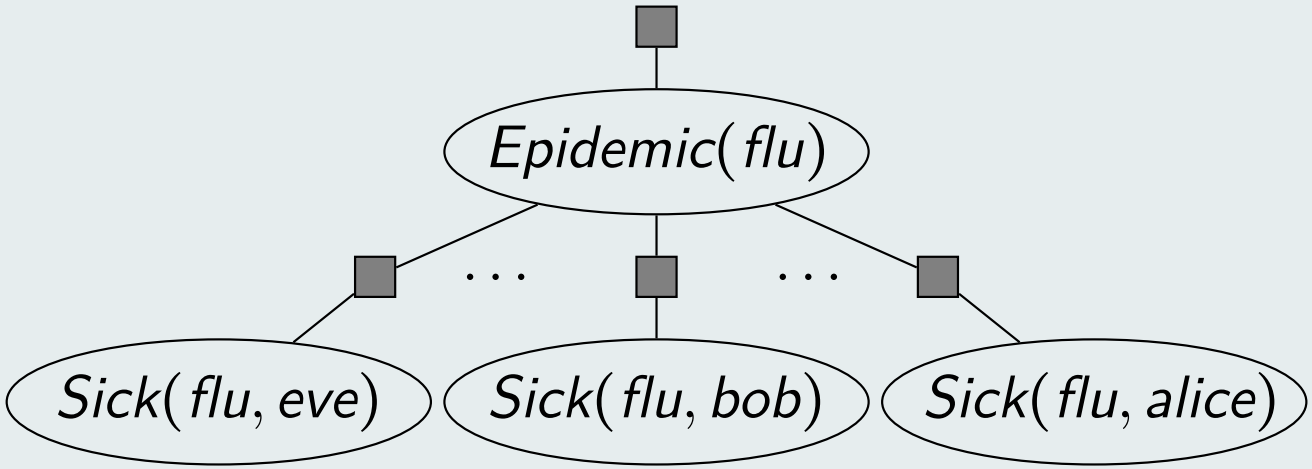
$\phi(Sick(flu, P), Epidemic(flu))$

Query

$P(Sick(flu, eve))$

(Lifted) Variable Elimination

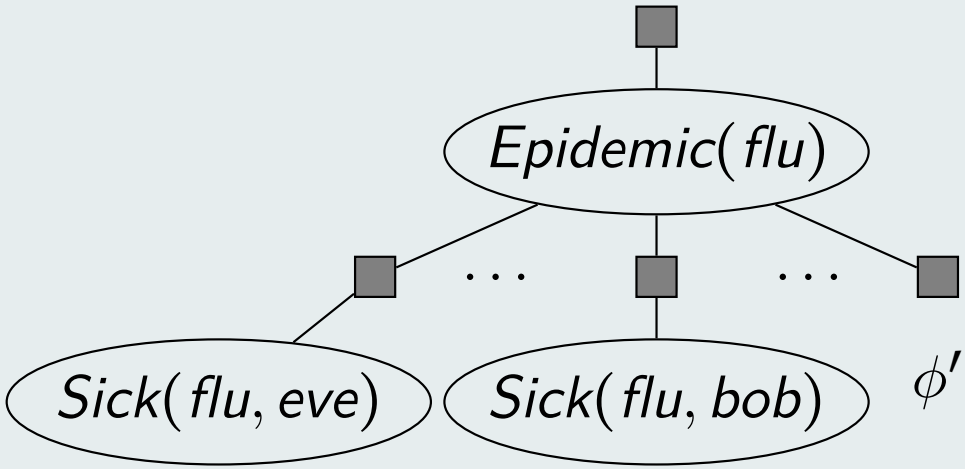
Zhang & Poole (1994), Poole (2003), de Salvo Braz (2007), Taghipour (2013)



$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e)$$
$$\cdot \ \ldots \ \cdot \sum_{sb \in range(Sick(flu,bob))} \phi(sb, e)$$
$$\cdot \ \ldots \ \cdot \sum_{sa \in range(Sick(flu,alice))} \phi(sa, e)$$

(Lifted) Variable Elimination

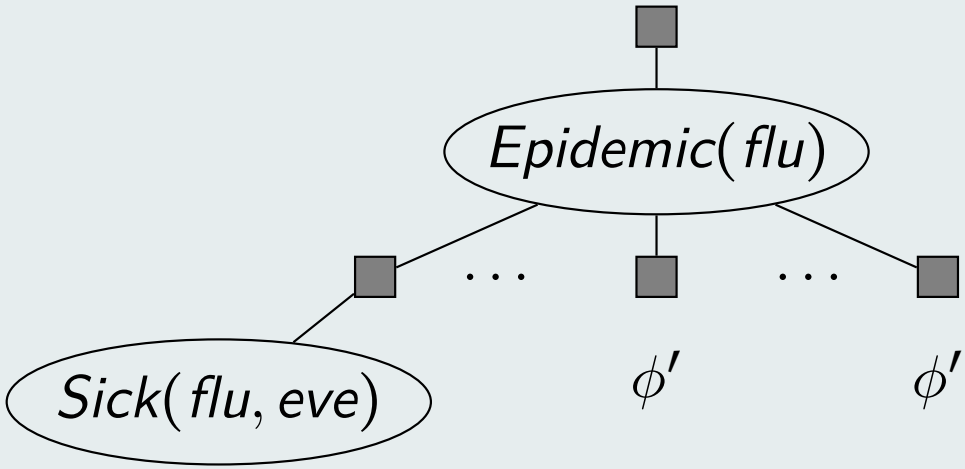
Zhang & Poole (1994), Poole (2003), de Salvo Braz (2007), Taghipour (2013)



$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e)$$
$$\cdot \ \ldots \ \cdot \sum_{sb \in range(Sick(flu,bob))} \phi(sb, e)$$
$$\cdot \ \ldots \ \cdot \ \phi'(e)$$

(Lifted) Variable Elimination

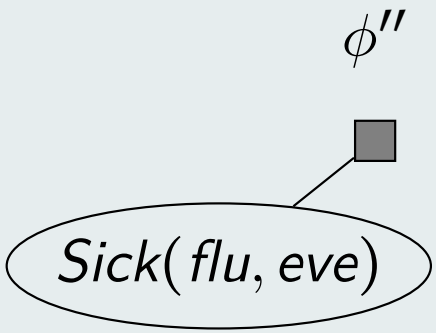
Zhang & Poole (1994), Poole (2003), de Salvo Braz (2007), Taghipour (2013)



$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e)$$
$$\cdot \ \ldots \ \cdot \ \phi'(e)$$
$$\cdot \ \ldots \ \cdot \ \phi'(e)$$

(Lifted) Variable Elimination

Zhang & Poole (1994), Poole (2003), de Salvo Braz (2007), Taghipour (2013)



$$P(Sick(flu, eve)) \propto \phi''(Sick(flu, eve))$$

(Lifted) Variable Elimination

Zhang & Poole (1994), Poole (2003), de Salzo Braz (2007), Taghipour (2013)

$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e)$$
$$\cdot \ \ldots \cdot \sum_{sb \in range(Sick(flu,bob))} \phi(sb, e)$$
$$\cdot \ \ldots \ \cdot \sum_{sa \in range(Sick(flu,alice))} \phi(sa, e)$$

(Lifted) Variable Elimination

Zhang & Poole (1994), Poole (2003), de Salzo Braz (2007), Taghipour (2013)

$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e) \cdot \ldots \cdot \sum_{sb \in range(Sick(flu,bob))} \phi(sb, e) \cdot \ldots \cdot \sum_{sa \in range(Sick(flu,alice))} \phi(sa, e)$$

$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e) \cdot \left(\sum_{s \in range(Sick(flu,P))} \phi(s, e) \right)^{|P|, P \neq eve}$$

(Lifted) Variable Elimination - Single Query

Zhang & Poole (1994), Poole (2003), de Salzo Braz (2007), Taghipour (2013)

$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e) \cdot \ldots \cdot \sum_{sb \in range(Sick(flu,bob))} \phi(sb, e) \cdot \ldots \cdot \sum_{sa \in range(Sick(flu,alice))} \phi(sa, e)$$

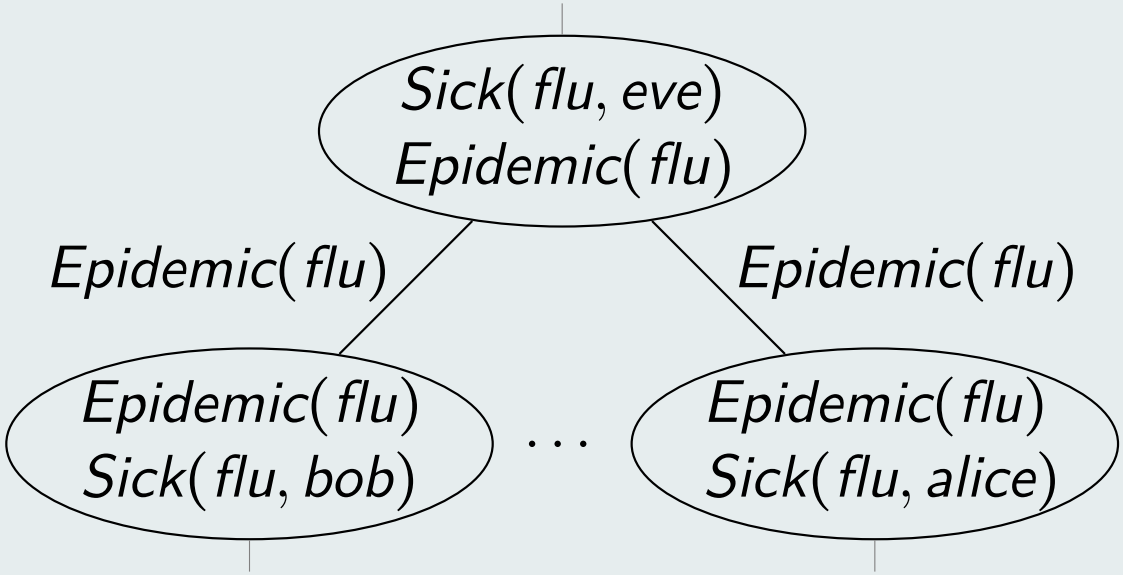
$$P(Sick(flu, eve)) \propto \sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e) \cdot \left(\sum_{s \in range(Sick(flu,P))} \phi(s, e) \right)^{|P|, P \neq eve}$$

Junction Trees - Multiple Queries

Shafer & Shenoy (1989), Jensen et. al. (1990), Darwiche (2009)

Internal Representation

$\phi(Sick(flu, eve), Epidemic(flu))\phi(Epidemic(flu))$



$\phi(Sick(flu, bob), Epidemic(flu))\phi(Sick(flu, alice), Epidemic(flu))$

Query

$P(Sick(flu, eve))$

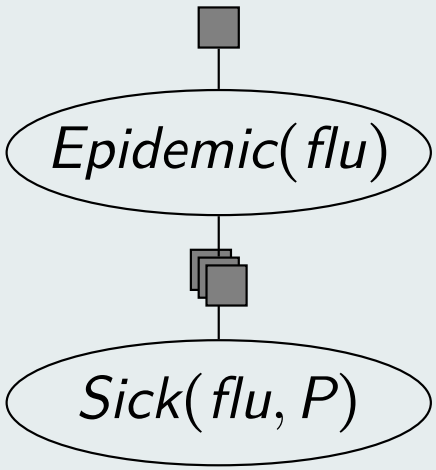
Computation

1. Construct junction tree.
2. Pass messages
   - inward
   - outward
3. Answer queries.

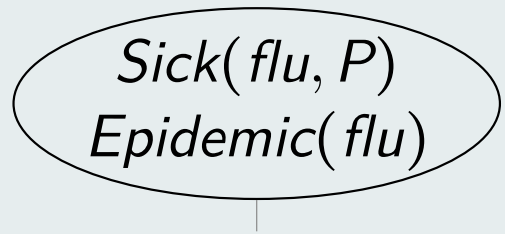
(without evidence)

Lifted Junction Tree Algorithm

Input Model



Internal Representation



$\phi(Sick(flu, P), Epidemic(flu))$

$\phi(Epidemic(flu))$

Query

$$P(Sick(flu, eve))$$

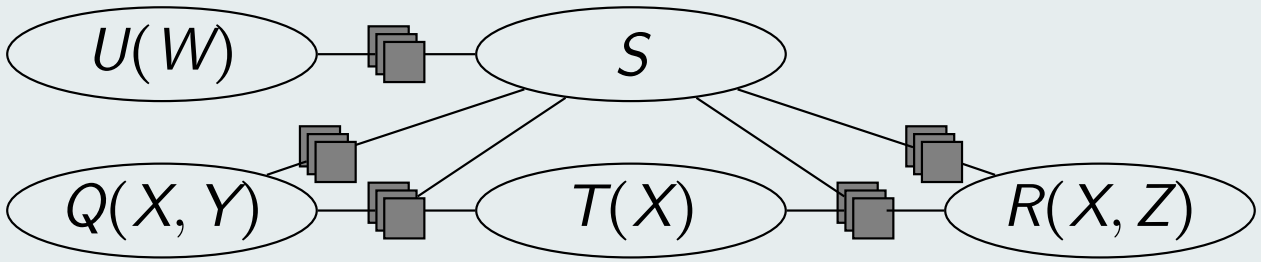
Computation

$$\sum_{e \in range(Epidemic(flu))} \phi(Sick(flu, eve), e) \cdot \phi(e) \cdot \left(\sum_{s \in range(Sick(flu,P))} \phi(s, e)\right)^{|P|, P \neq eve}$$

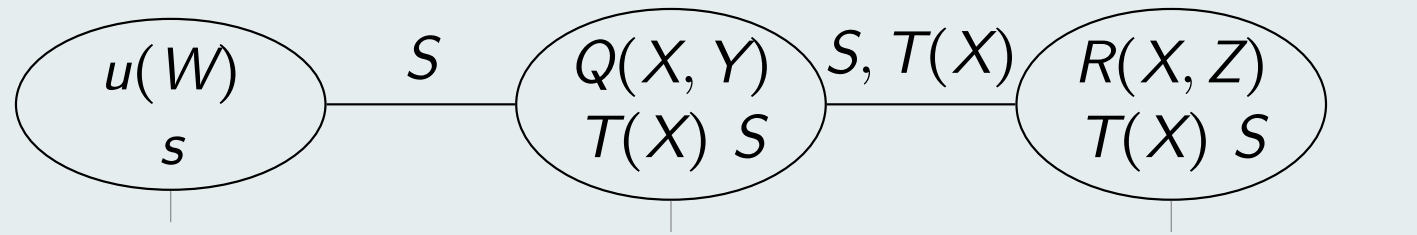
Lifted Junction Tree Algorithm

Another Example

Input Model



Internal Representation



$\phi_1 = \phi(U(W), S)$

$\phi_2 = \phi(Q(X,Y), T(X), S)$ $\phi_3 = \phi(R(X,Z), T(X), S)$

$\phi(Q(X,Y), S)$ $\phi(R(X,Z), S))$

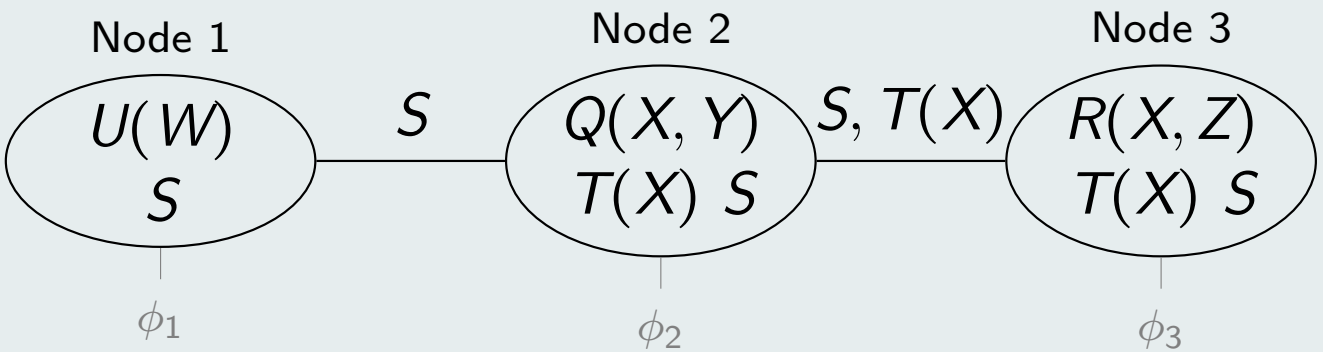
Queries

$P(Q(x_1, y_1))$ $P(U(w_1))$ $P(T(x_1))$

Lifted Junction Tree Algorithm

Message Passing

Internal Representation



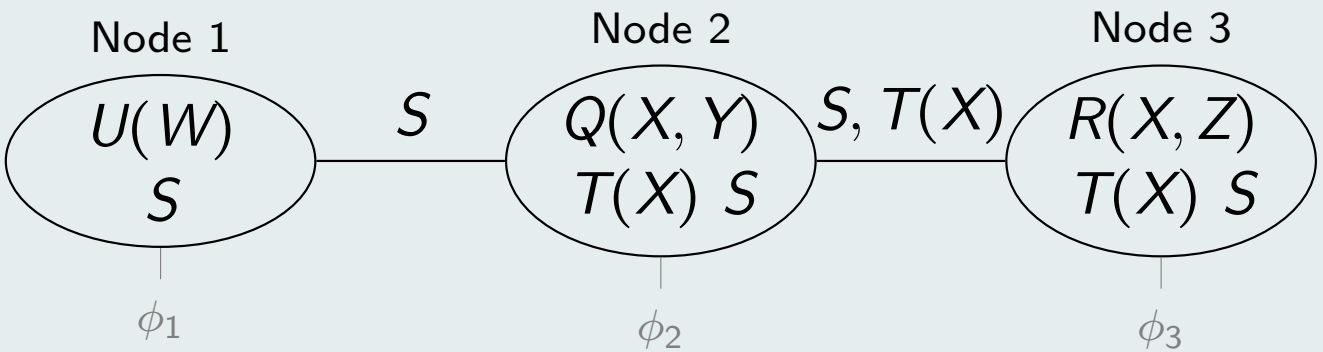
Queries

$$P(Q(x_1, y_1))$$
$$P(U(w_1))$$
$$P(T(x_1))$$

Lifted Junction Tree Algorithm

Message Passing

Internal Representation



Queries

$P(Q(x_1, y_1))$

$P(U(w_1))$

$P(T(x_1))$

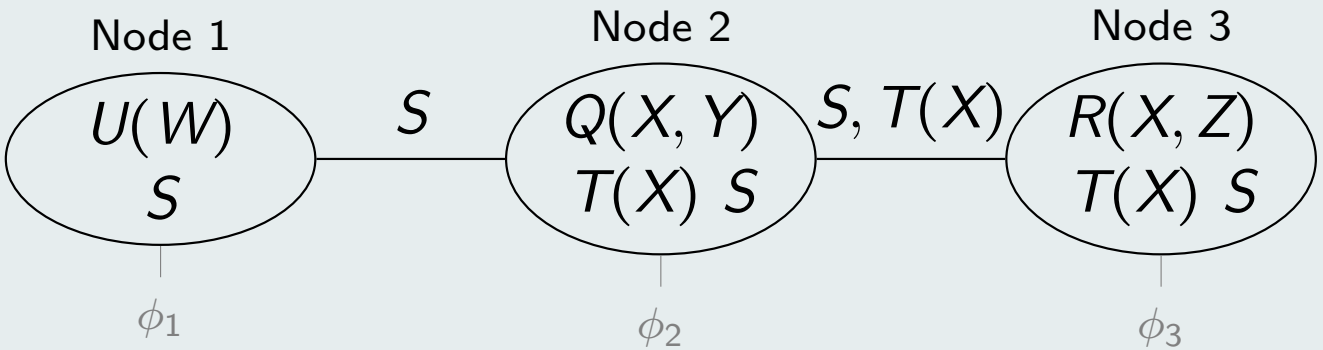
Inward pass

- From node 1
  - to node 2: $m_{12} = \sum_{U(W)} \phi_1$
- From node 3
  - to node 2: $m_{32} = \sum_{R(X,Z)} \phi_3$

Lifted Junction Tree Algorithm

Message Passing

Internal Representation



Queries

$P(Q(x_1, y_1))$

$P(U(w_1))$

$P(T(x_1))$

Inward pass

- From node 1
  - to node 2: $m_{12} = \sum_{U(W)} \phi_1$
- From node 3
  - to node 2: $m_{32} = \sum_{R(X,Z)} \phi_3$

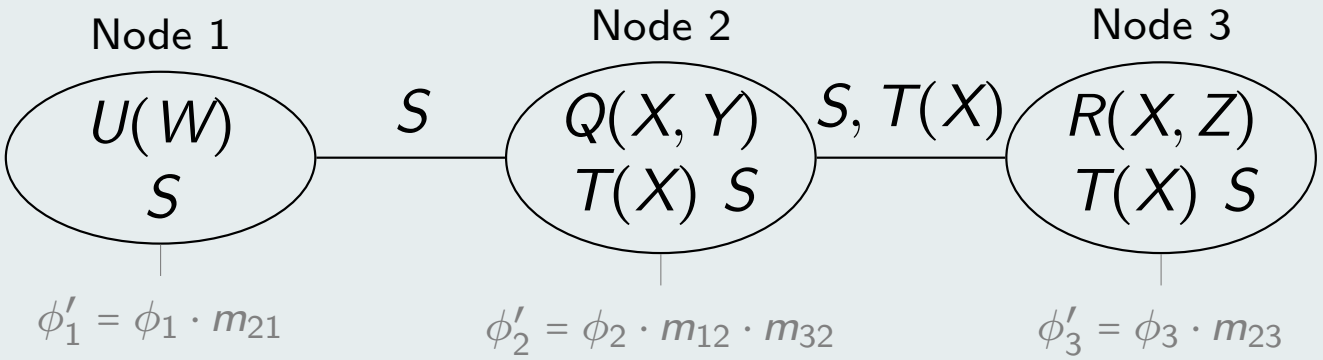
Outward pass

- From node 2
  - to node 1: $m_{21} = \sum_{T(X)} \sum_{Q(X,Z)} \phi_2 \cdot m_{32}$
  - to node 3: $m_{23} = \sum_{Q(X,Z)} \phi_2 \cdot m_{12}$

Lifted Junction Tree Algorithm

Query Answering

Internal Representation



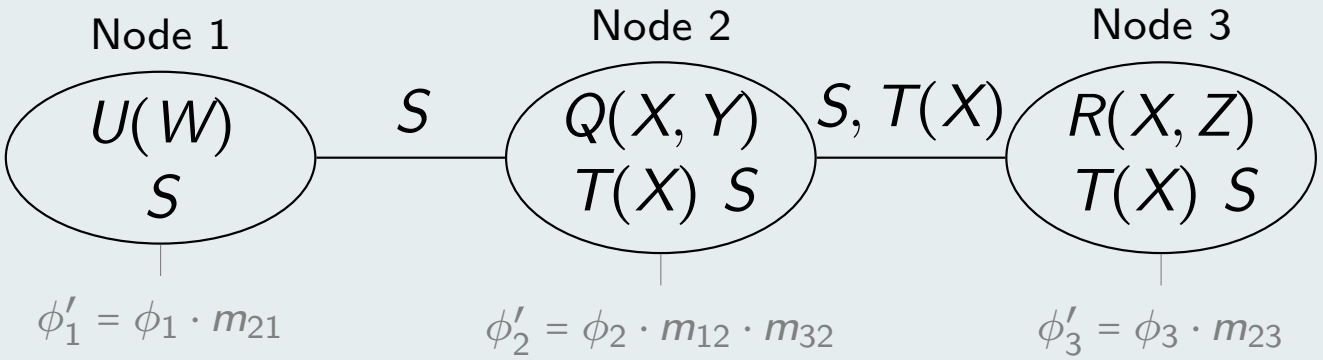
Queries

$$P(Q(x_1, y_1))$$
$$P(U(w_1))$$
$$P(T(x_1))$$

Lifted Junction Tree Algorithm

Query Answering

Internal Representation



Queries

$$P(Q(x_1, y_1))$$
$$P(U(w_1))$$
$$P(T(x_1))$$

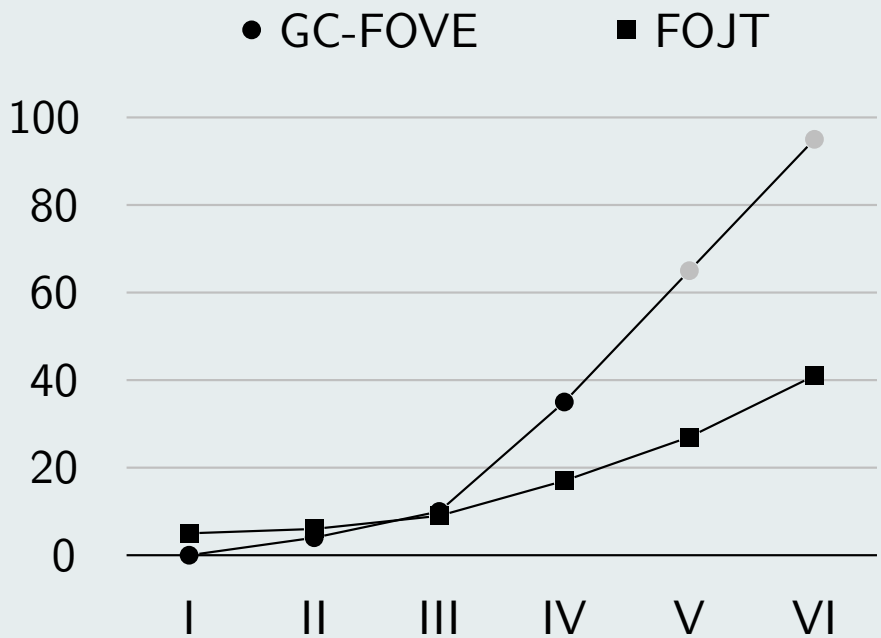
Computation

$$P(Q(x_1, y_1)) \propto \sum_{S} \sum_{\substack{T(X) \\ X \neq x_1}} \sum_{\substack{Q(X,Y) \\ X \neq x_1 \\ Y \neq y_1}} \phi_2'$$

Analogously, for queries $P(U(w_1))$ and $P(T(x_1))$.

Test Run and First Results

Comparison with GC-FOVE, Implementation of Lifted VE by Taghipour (2013)



I	message passing
II	$P(S) + \text{I}$
III	$P(U(w_1)) + \text{II}$
IV	$P(T(x_1)) + \text{III}$
V	$P(Q(x_1, y_1)) + \text{IV}$
VI	$P(R(x_1, z_1)) + \text{V}$

Figure: Accumulated Number of Calls to Lifted VE Operators during Query Answering by GC-FOVE and FOJT
Points connected for readability. Grey dot denotes count includes aborted queries.

Take Home Message

Research Problem

Optimise answering multiple queries

Proposed Solution

Lifted Junction Tree Algorithm

Take Home Message

Research Problem

Optimise answering multiple queries

Proposed Solution

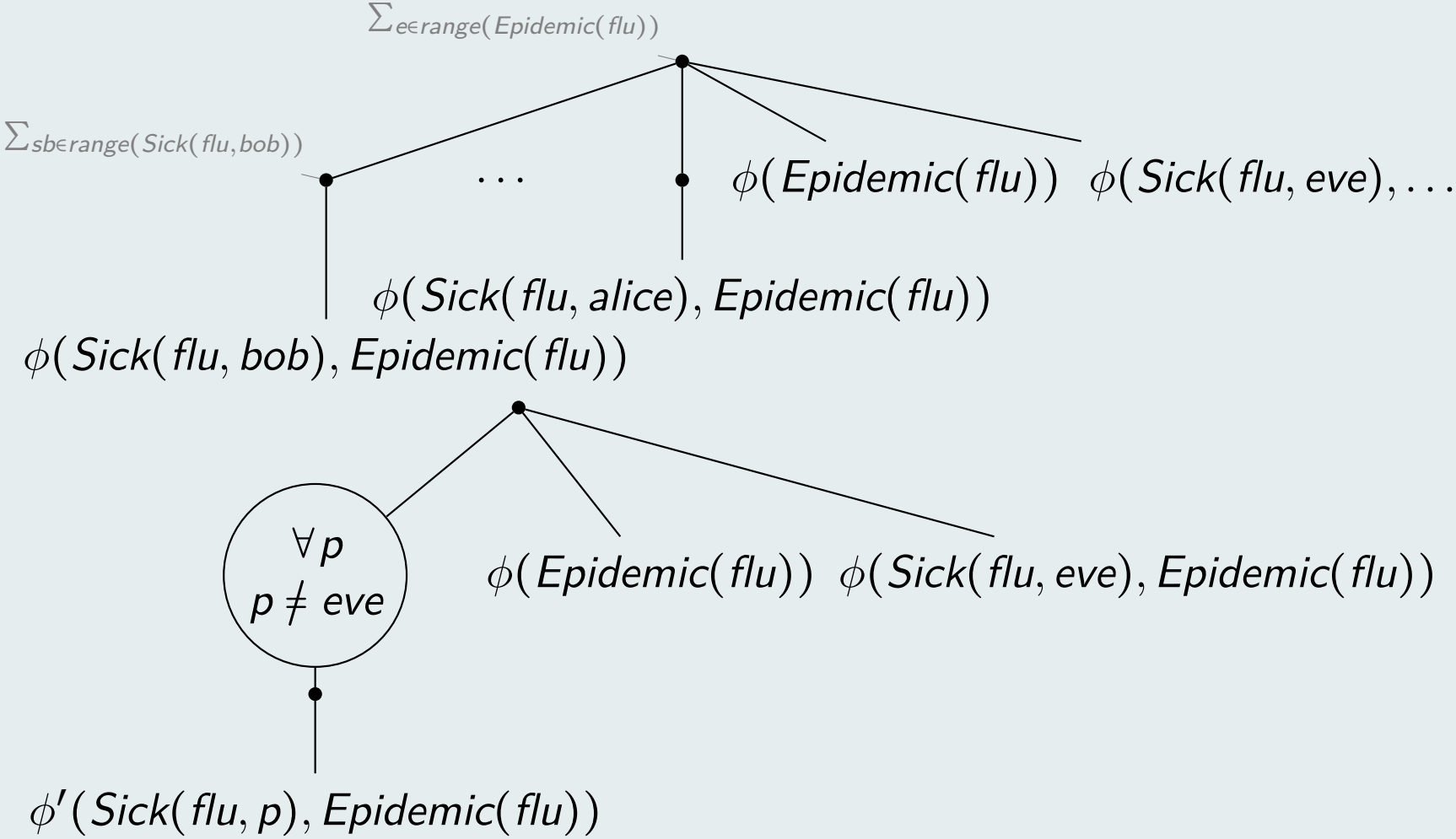
Lifted Junction Tree Algorithm

Future Directions

- Local symmetries?
- Dynamic variants?

(First-order) Decomposition Trees

Darwiche (2001), Taghipour (2013)



Cutset, Context, Cluster

Darwiche (2001), Taghipour (2013)

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

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

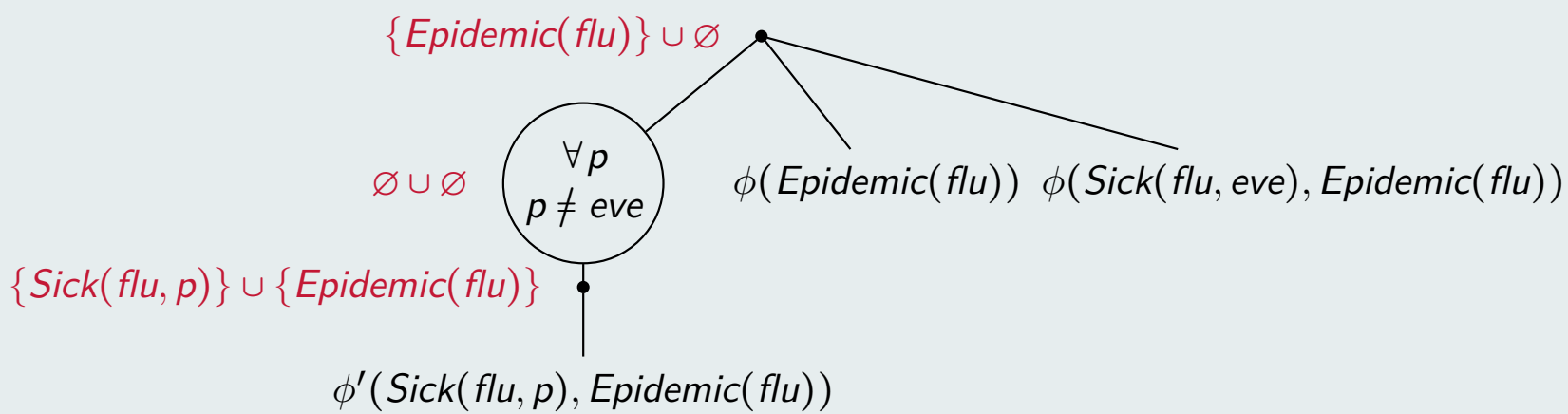
$$context(T) = RV(T) \cap acutset(T)$$

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

Random variables:

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

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



Labels to the left: $cutset(N) \cup context(N)$