Adaptive Inference on Probabilistic Relational Models

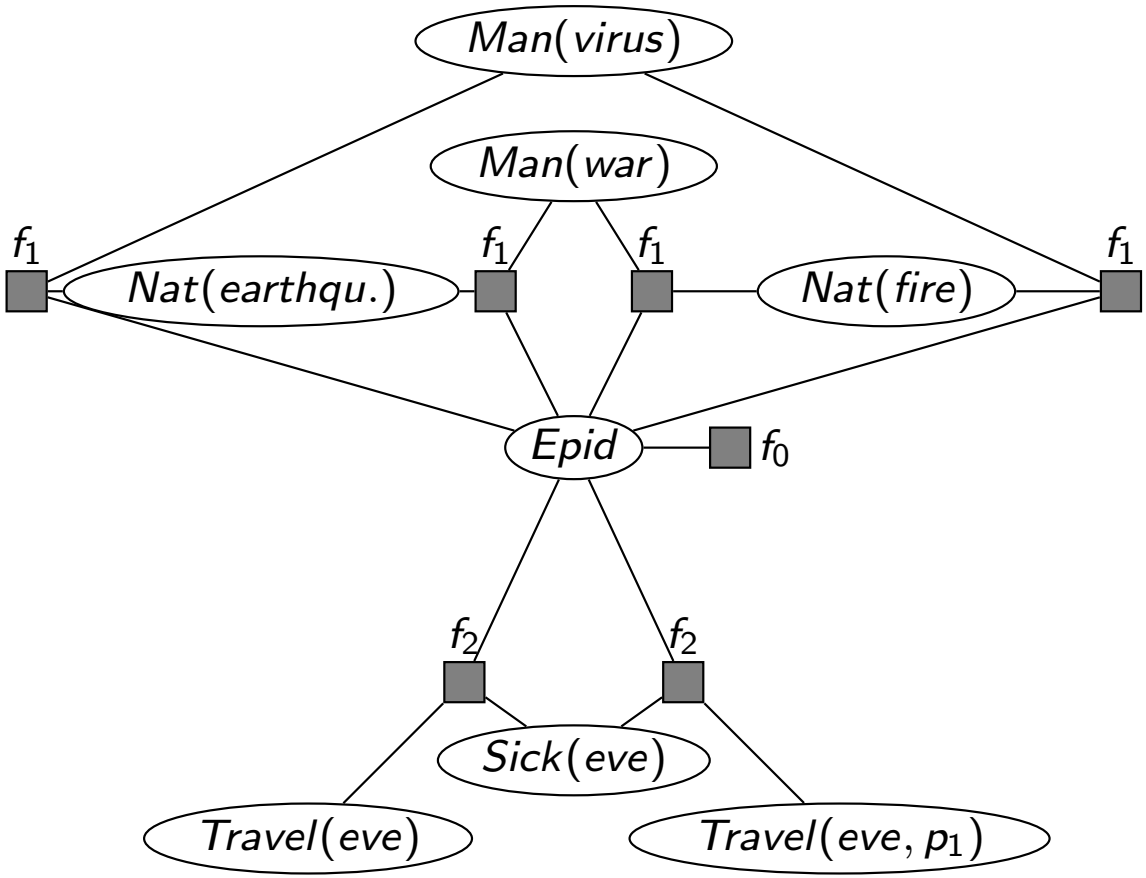
Tanya Braun, Ralf Möller

Institute of Information Systems
University of Lübeck

December 12, 2018

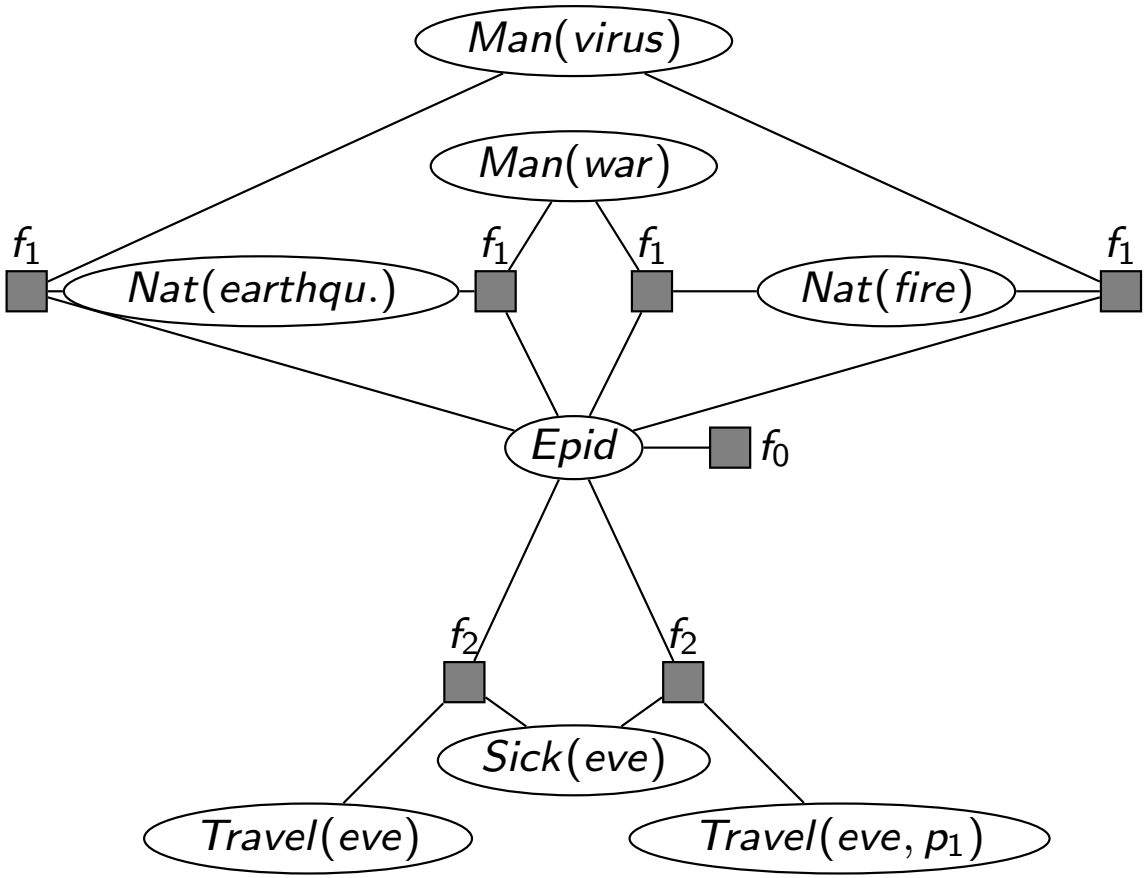
Probabilistic Graphical Models

Factor Graph F: Compact Encoding of Full Joint Distribution $P_F = \frac{1}{Z} \prod_i f_i$



Probabilistic Graphical Models

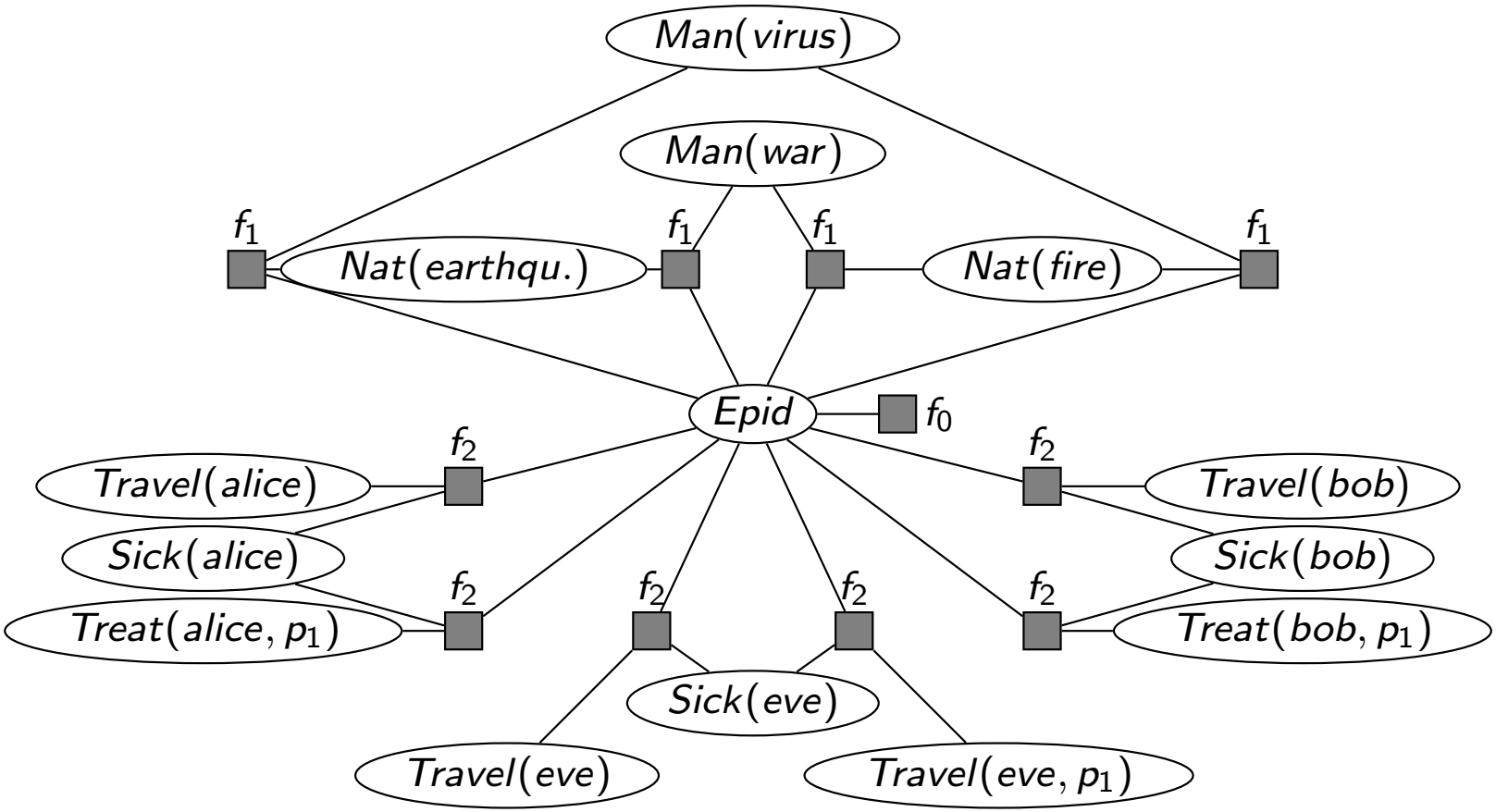
Factor Graph F: Compact Encoding of Full Joint Distribution $P_F = \frac{1}{Z} \prod_i f_i$



Query answering (QA): Eliminate all non-query variables
avoiding building P_F

Probabilistic Graphical Models

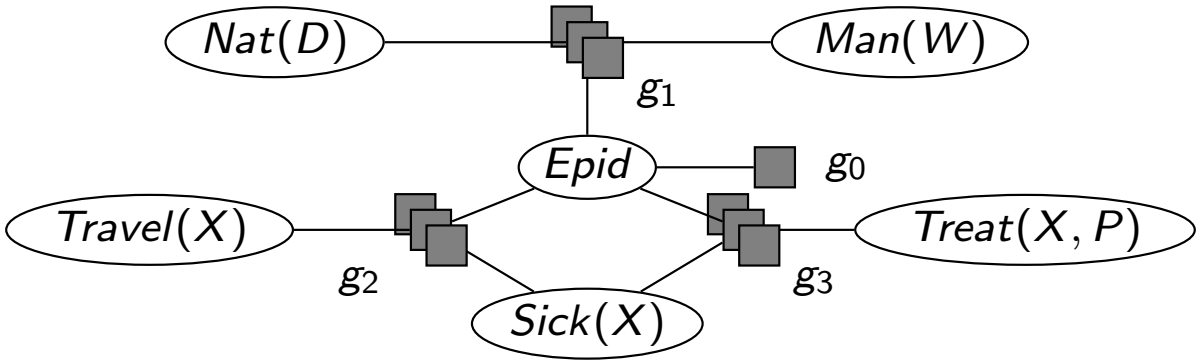
Factor Graph F: Compact Encoding of Full Joint Distribution $P_F = \frac{1}{Z} \prod_i f_i$



Query answering (QA): Eliminate all non-query variables
avoiding building P_F

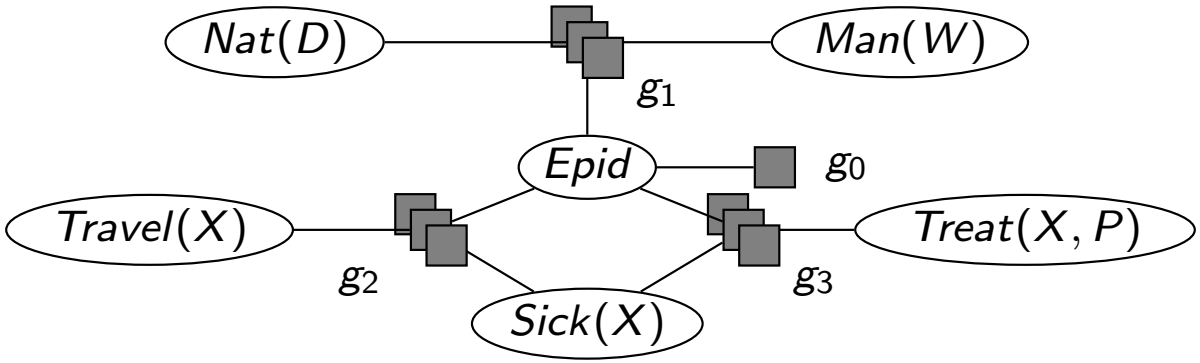
Probabilistic Relational and Lifted Models

Parfactor Graph G: Compact Encoding of Full Joint Distribution $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$



Probabilistic Relational and Lifted Models

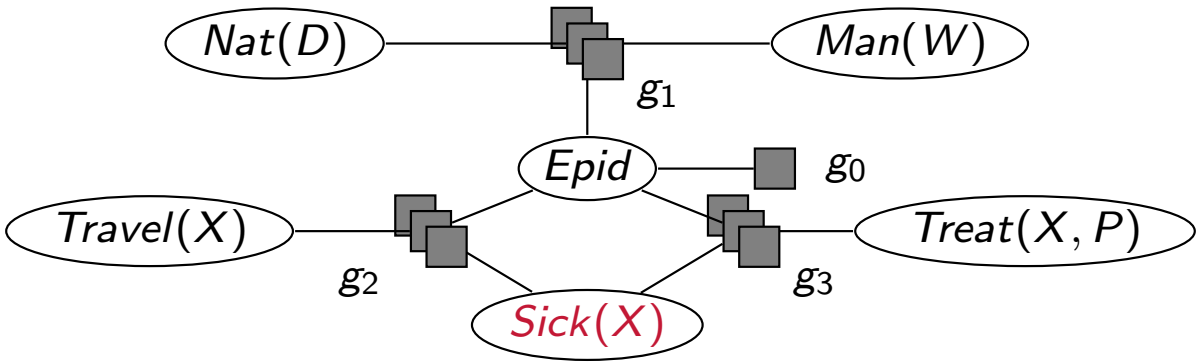
Parfactor Graph G: Compact Encoding of Full Joint Distribution $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$



QA: Eliminate all non-query variables
while **avoiding building P_F and avoiding groundings**

QA: Lifted Variable Elimination (LVE)

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

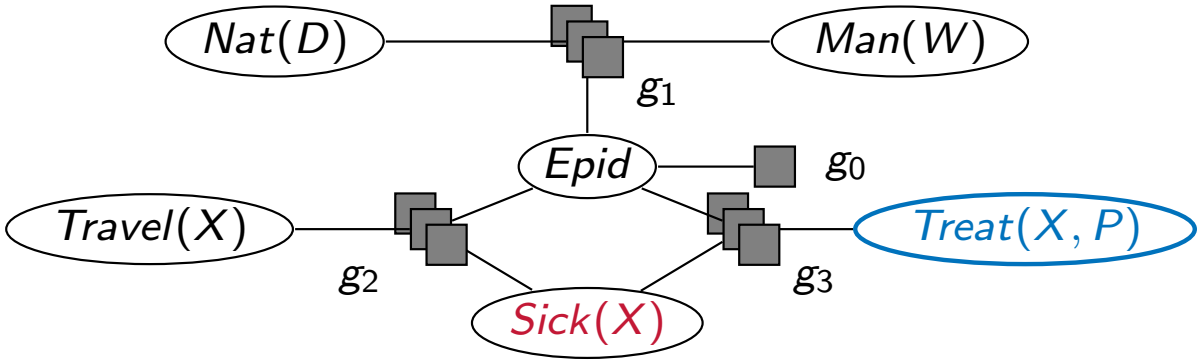


$P(Sick(eve)) \propto$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

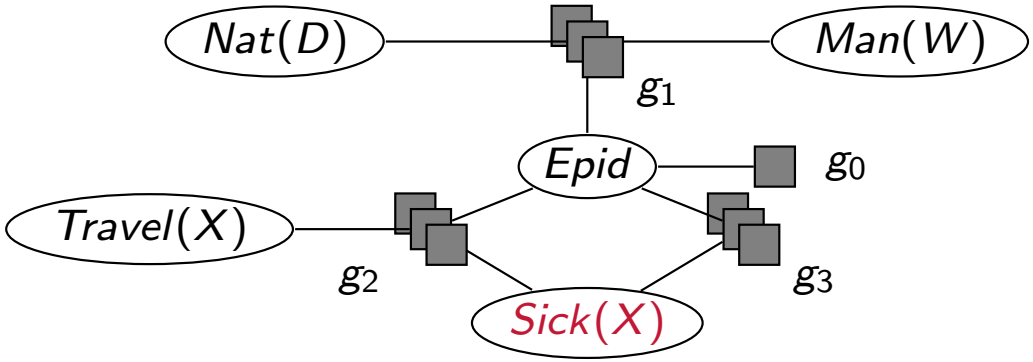


$$P(Sick(eve)) \propto \sum_{Treat(X,P)} g_3$$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

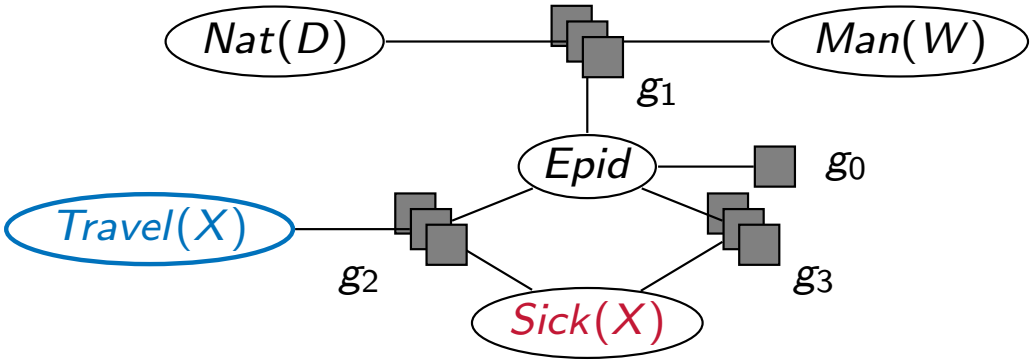


$$P(Sick(eve)) \propto \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|}$$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

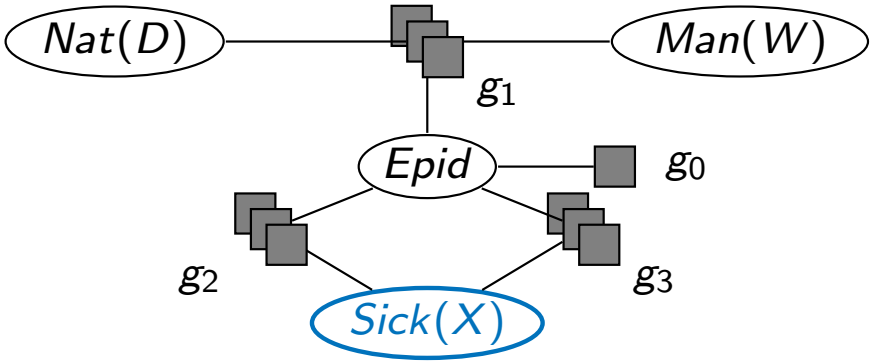


$$P(Sick(eve)) \propto \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|}$$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

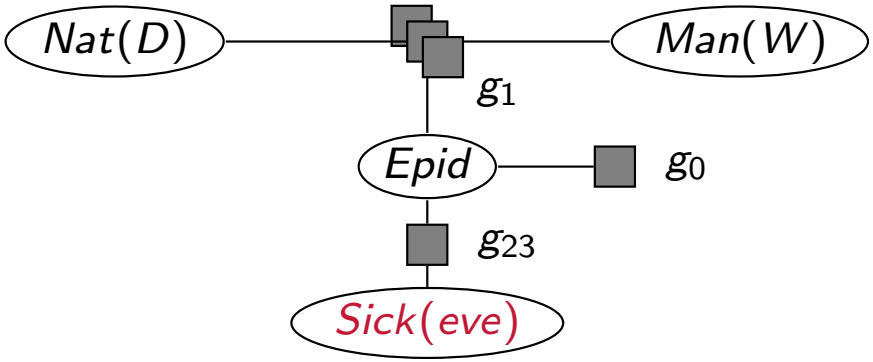


$$P(Sick(eve)) \propto \sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|}$$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

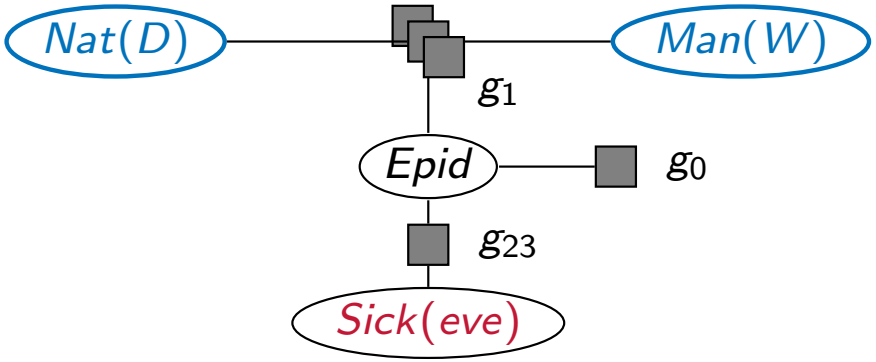


$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

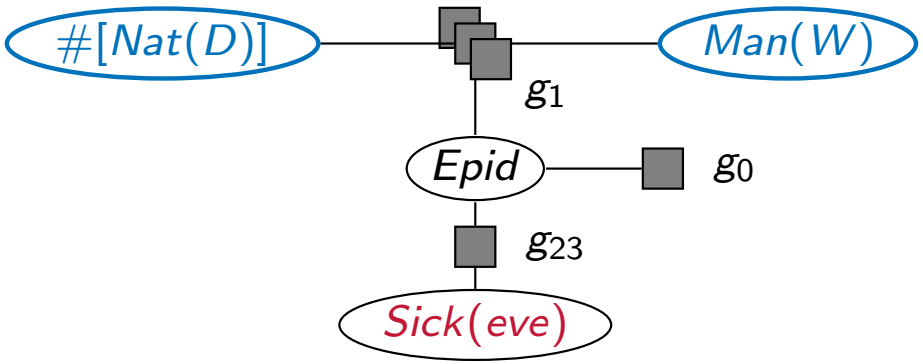


$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|}_{X \neq eve}$$

$\sum_V$ indicates a sum over the values of V; # indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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



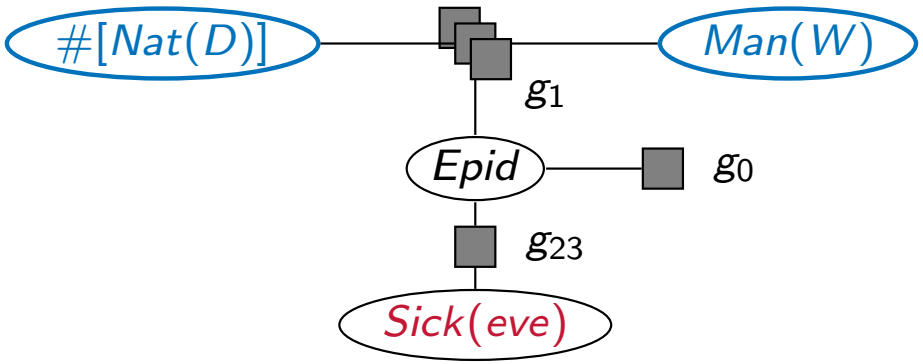
$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}}$$

$$g_1^{\#}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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



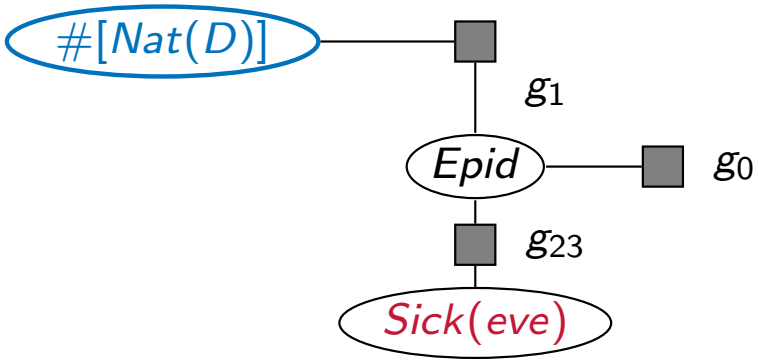
$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}}$$

$$\sum_{Man(W)} g_1^{\#}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

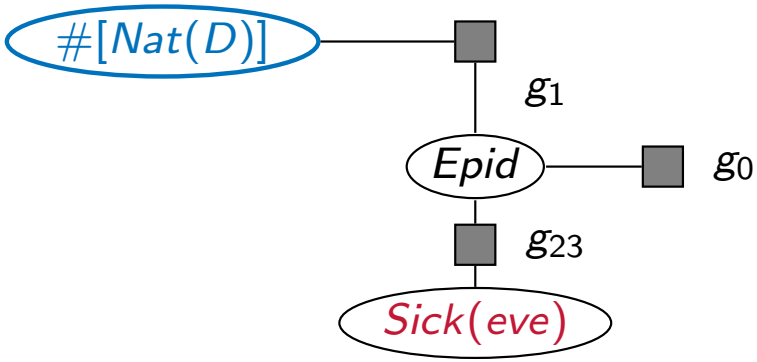


$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}} \left(\sum_{Man(W)} g_1^{\#} \right)^{|dom(W)|}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

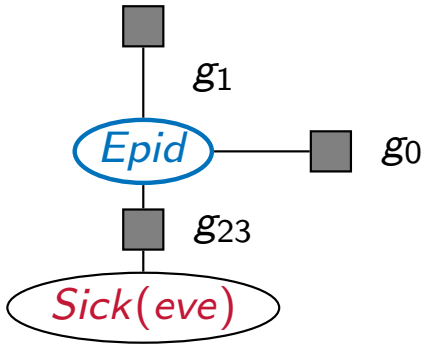


$$P(Sick(eve)) \propto \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}} \sum_{\#_D[Nat(D)]} \left(\sum_{Man(W)} g_1^{\#} \right)^{|dom(W)|}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

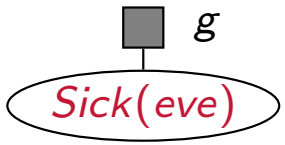


$$P(Sick(eve)) \propto \sum_{Epid} g_0 \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}} \sum_{\#_D[Nat(D)]} \left(\sum_{Man(W)} g_1^{\#} \right)^{|dom(W)|}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

QA: Lifted Variable Elimination (LVE)

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

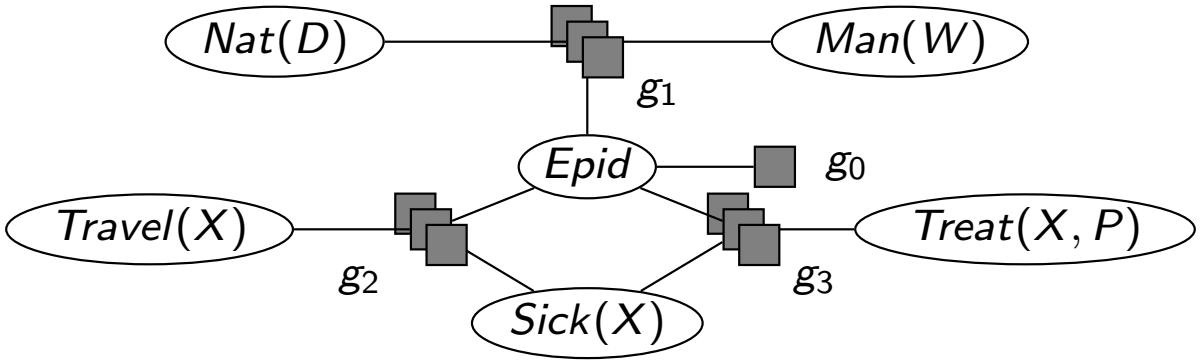


$$P(Sick(eve)) \propto \sum_{Epid} g_0 \left(\sum_{\substack{Sick(X) \\ X \neq eve}} \sum_{Travel(X)} g_2 \left(\sum_{Treat(X,P)} g_3 \right)^{|dom(P)|} \right)^{|dom(X)|_{X \neq eve}} \sum_{\#_D[Nat(D)]} \left(\sum_{Man(W)} g_1^{\#} \right)^{|dom(W)|}$$

$\sum_V$ indicates a sum over the values of V; $\#$ indicates a count conversion

Repeated Inference

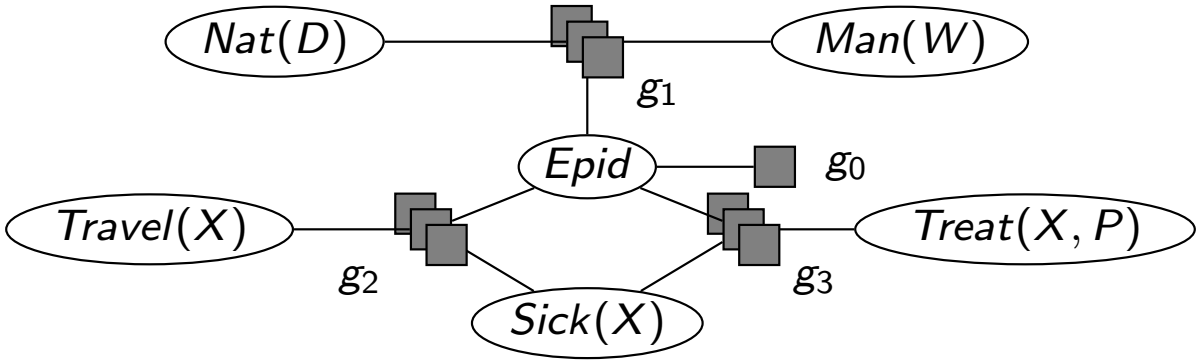
Answer Many Queries for Marginal Distributions



Solve each query more efficiently than starting from scratch for each query

Repeated Inference

Answer Many Queries for Marginal Distributions



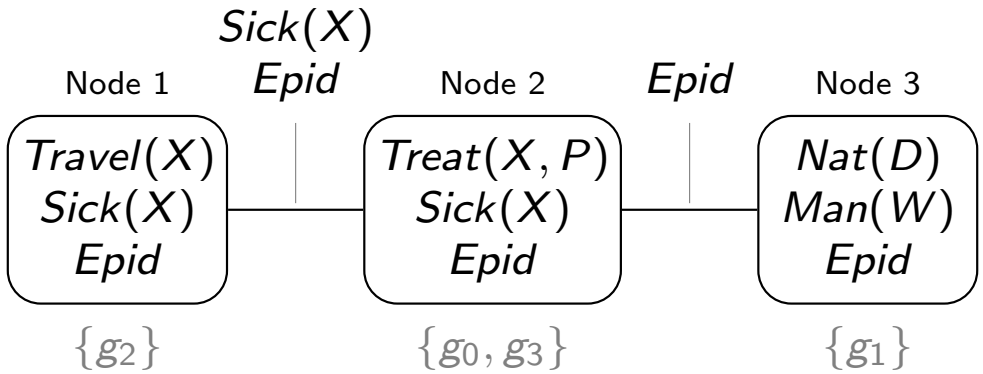
Solve each query more efficiently than starting from scratch for each query

General Idea

1. Build a helper structure.
2. Precompute as much computations as possible, including evidence **E**.
3. Using the precomputed information, answer individual queries.

Lifted Junction Tree Algorithm (LJT)

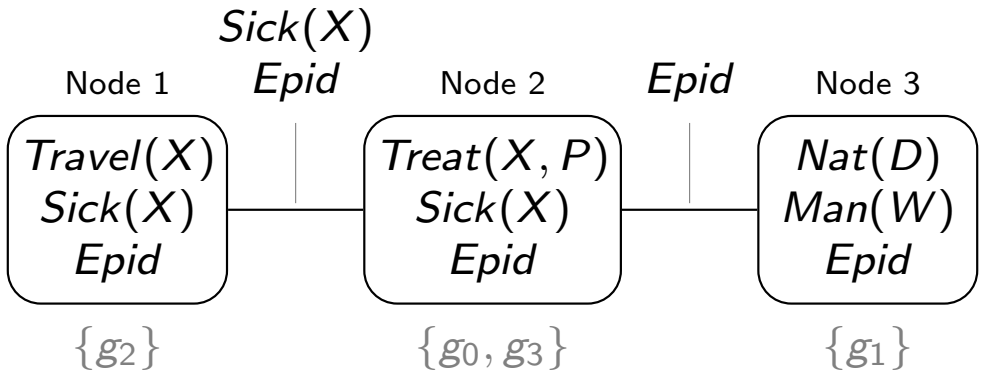
Braun and Möller (2016)



❶ LJT constructs a first-order junction tree for model

Lifted Junction Tree Algorithm (LJT)

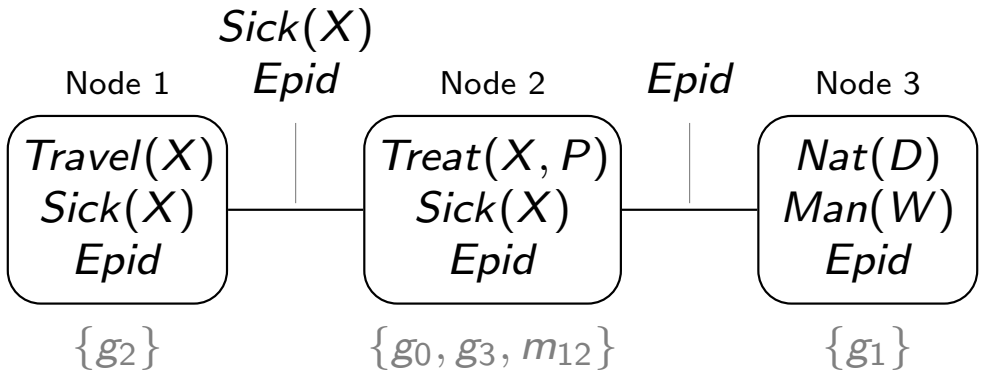
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
   - Message = query on edge variables

Lifted Junction Tree Algorithm (LJT)

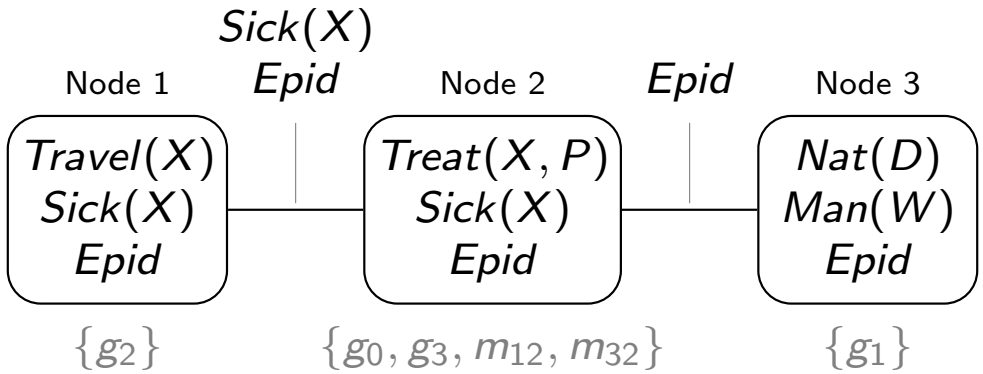
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
   - Message = query on edge variables
   - 1 $\longrightarrow$ 2: $P(Sick(X), Epid)$
     - Eliminate $Travel(X)$

Lifted Junction Tree Algorithm (LJT)

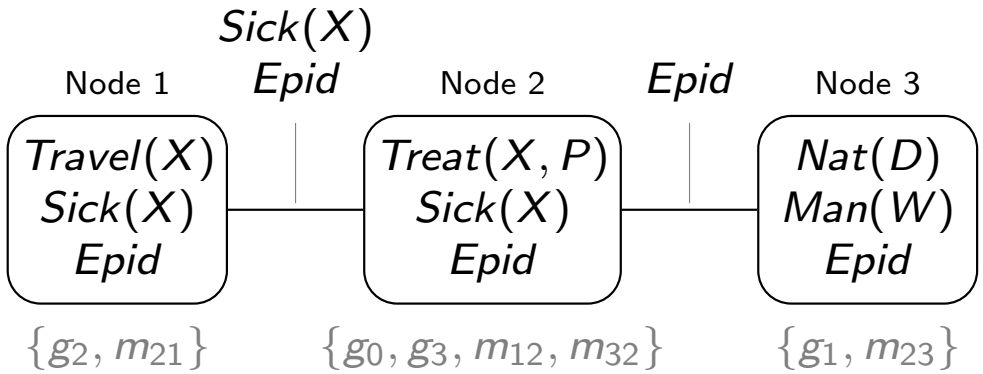
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
   - Message = query on edge variables
   - 1 $\longrightarrow$ 2: $P(Sick(X), Epid)$
     - Eliminate $Travel(X)$
   - 3 $\longrightarrow$ 2: $P(Epid)$
     - Eliminate $Nat(D)$, $Man(W)$

Lifted Junction Tree Algorithm (LJT)

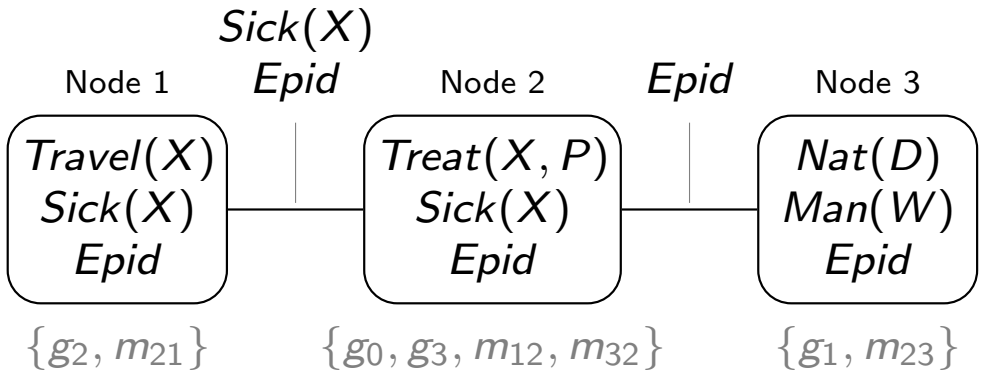
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
   - Message = query on edge variables
   - 1 $\longrightarrow$ 2: $P(Sick(X), Epid)$
     - Eliminate $Travel(X)$
   - 3 $\longrightarrow$ 2: $P(Epid)$
     - Eliminate $Nat(D)$, $Man(W)$
   - Messages from node 2

Lifted Junction Tree Algorithm (LJT)

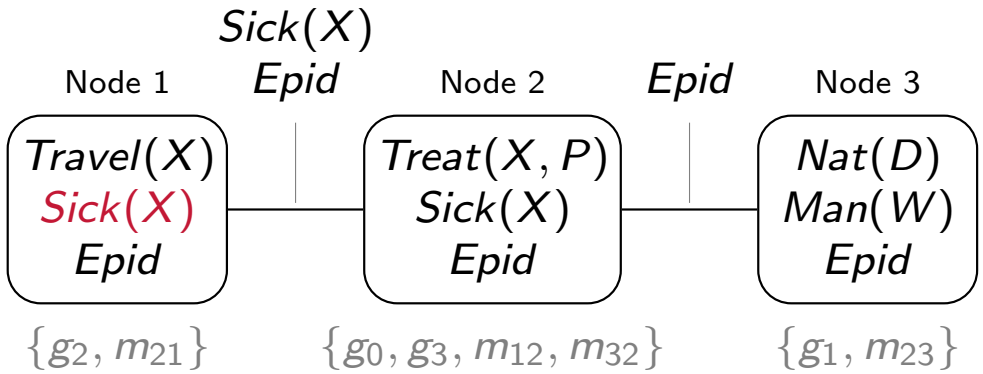
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
3. LJT answers queries
   - using node where query term appears
   - reusing messages, avoiding repeated calculations

Lifted Junction Tree Algorithm (LJT)

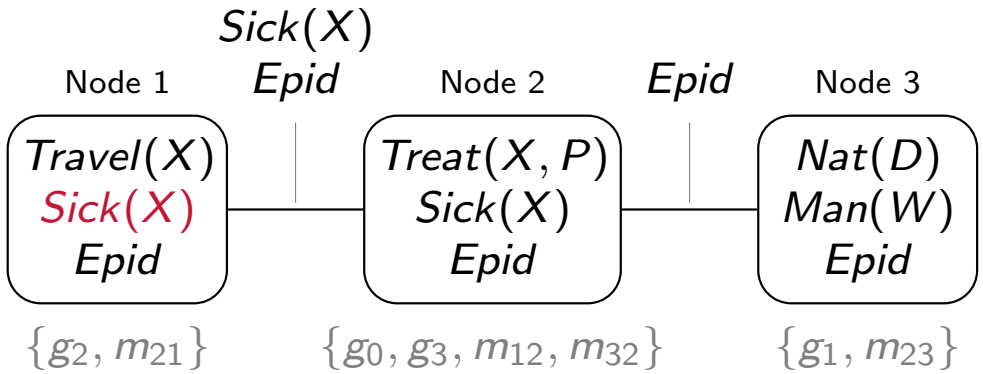
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
3. LJT answers queries
   - using node where query term appears
   - reusing messages, avoiding repeated calculations
   - $P(Sick(eve))$: node 1 with local model g_2, m_{12}

Lifted Junction Tree Algorithm (LJT)

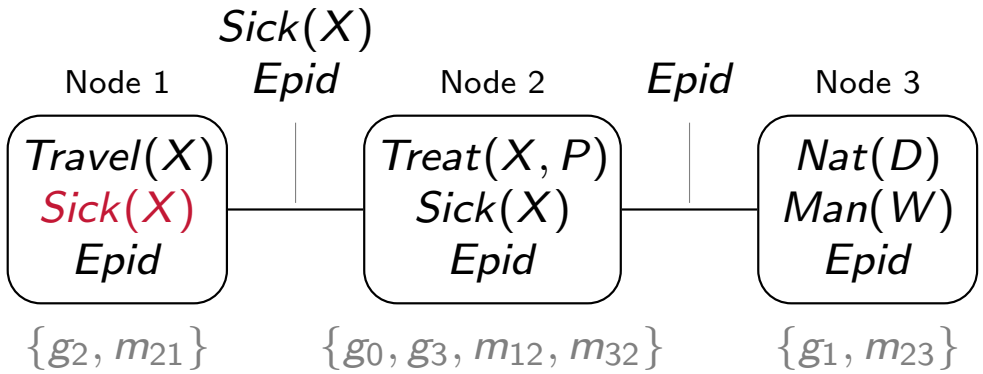
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
3. LJT answers queries
   - using node where query term appears
   - reusing messages, avoiding repeated calculations
   - $P(Sick(eve))$: node 1 with local model g_2, m_{12}
     - Eliminate $Travel(X)$ from g_2

Lifted Junction Tree Algorithm (LJT)

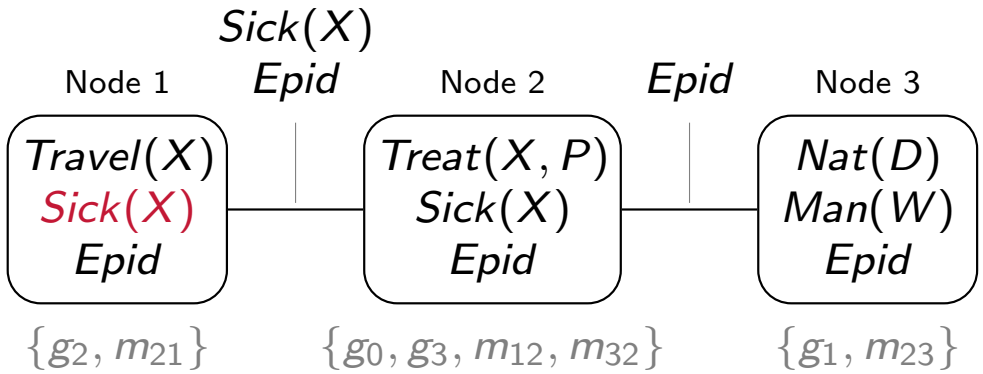
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
3. LJT answers queries
   - using node where query term appears
   - reusing messages, avoiding repeated calculations
   - $P(Sick(eve))$: node 1 with local model g_2, m_{12}
     - Eliminate $Travel(X)$ from g_2
     - Eliminate $Sick(X)$, $X \neq eve$, from $g_2' \cdot m_{21}$

Lifted Junction Tree Algorithm (LJT)

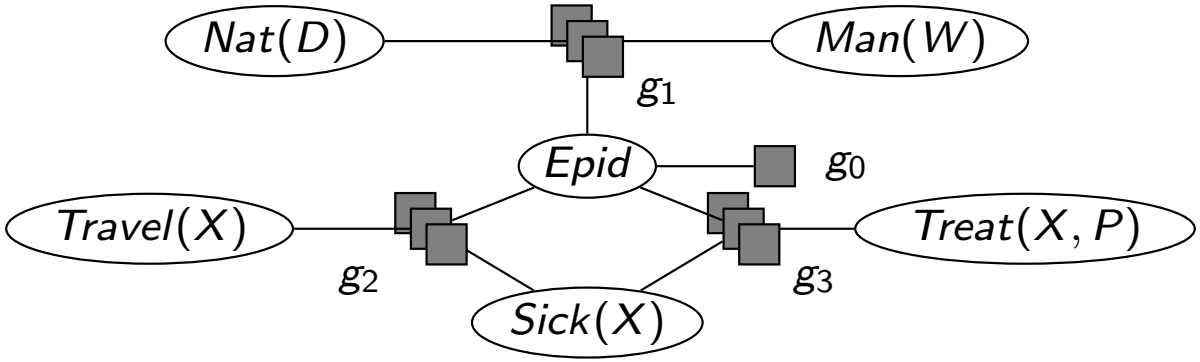
Braun and Möller (2016)



1. LJT constructs a first-order junction tree for model
2. LJT passes messages (independent of query)
3. LJT answers queries
   - using node where query term appears
   - reusing messages, avoiding repeated calculations
   - $P(Sick(eve))$: node 1 with local model g_2, m_{12}
     - Eliminate $Travel(X)$ from g_2
     - Eliminate $Sick(X)$, $X \neq eve$, from $g_2' \cdot m_{21}$
     - Eliminate $Epid$ from g_{21}'

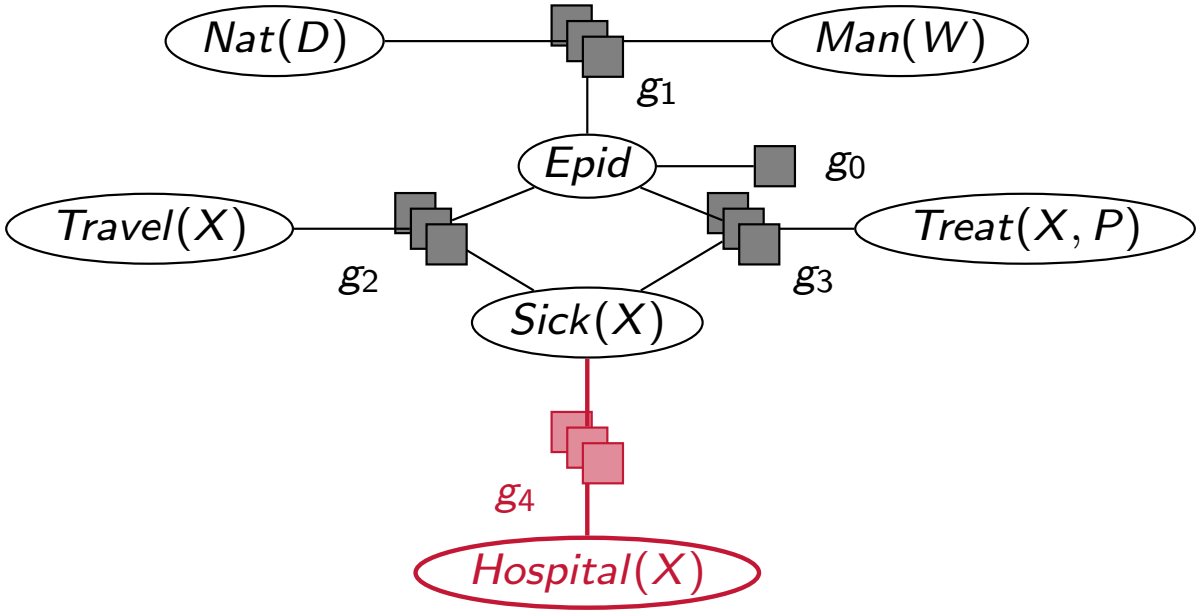
Adaptive Inference

Answer Queries for Marginal Distributions in an Incrementally Changing Model



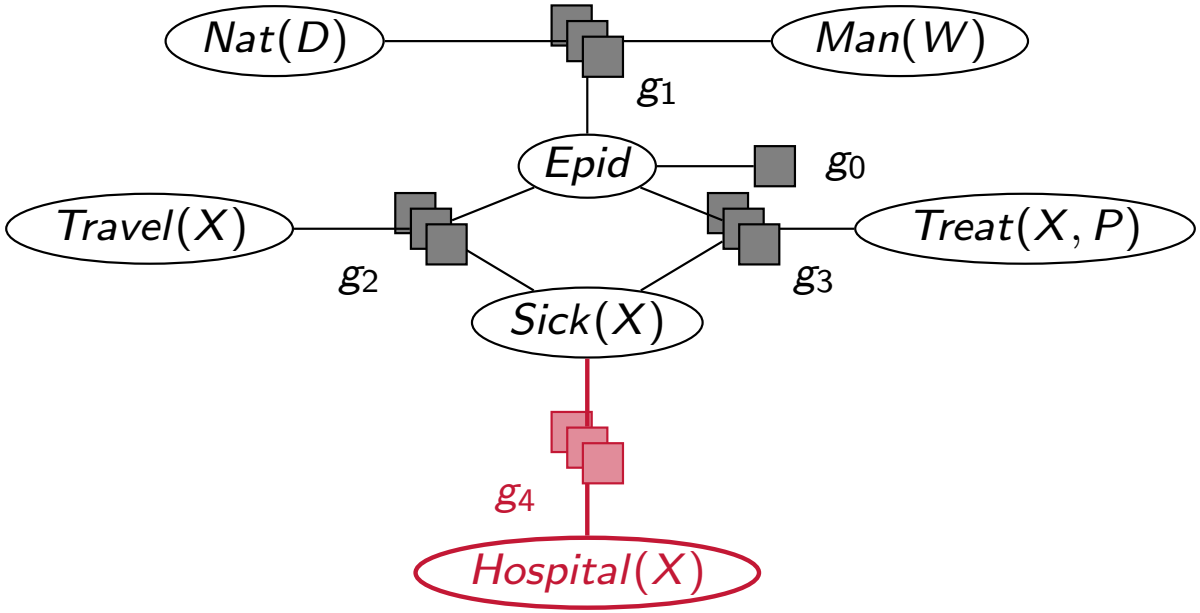
Adaptive Inference

Answer Queries for Marginal Distributions in an Incrementally Changing Model



Adaptive Inference

Answer Queries for Marginal Distributions in an Incrementally Changing Model

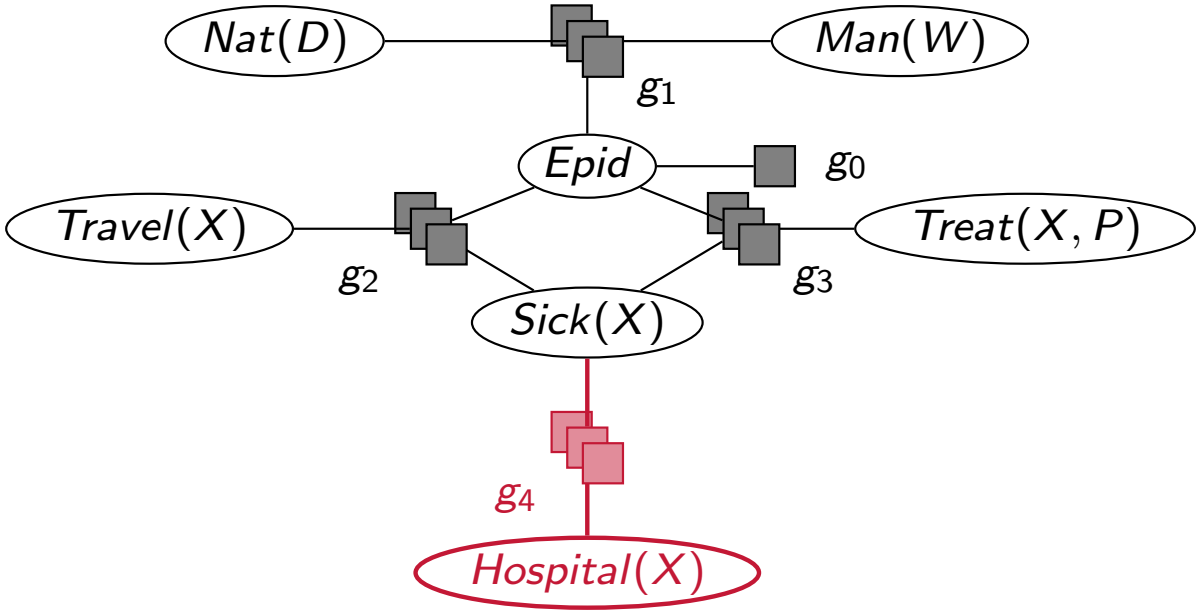


Evidence changes (new/retracted)

$$Sick(alice) = true$$
$$Sick(bob) = true$$

Adaptive Inference

Answer Queries for Marginal Distributions in an Incrementally Changing Model



Evidence changes (new/retracted)

$Sick(alice) = true$

$Sick(bob) = true$

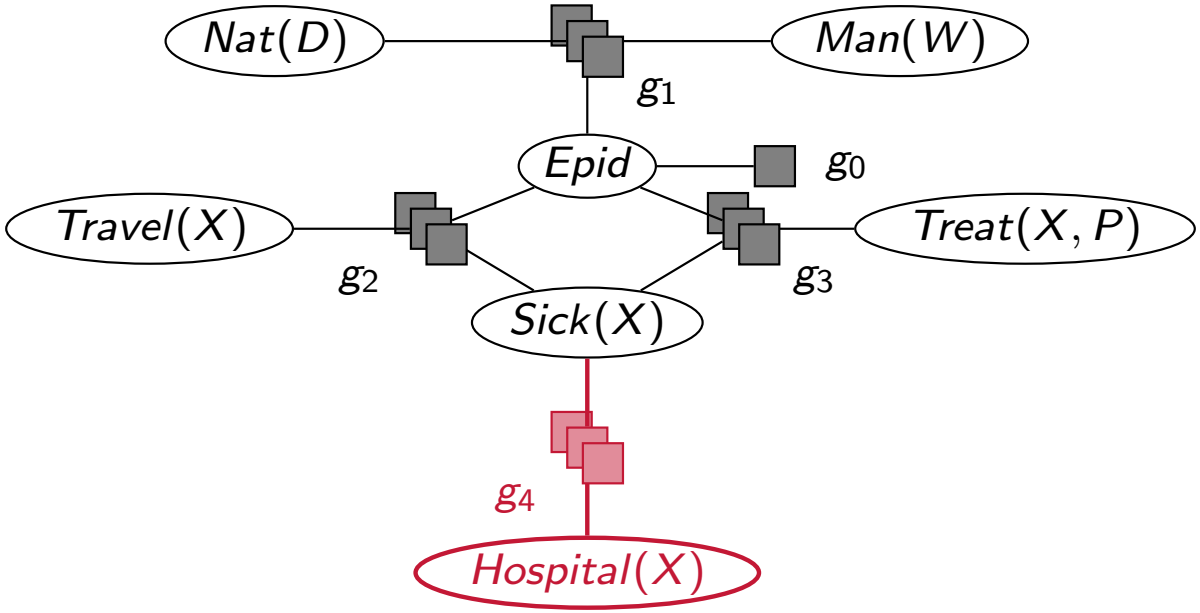
Domains change (new constants)

$X \in \{alice, eve, bob\}$

$\cup\{charlie\}$

Adaptive Inference

Answer Queries for Marginal Distributions in an Incrementally Changing Model



Evidence changes (new/retracted)

$Sick(alice) = true$

$Sick(bob) = true$

Domains change (new constants)

$X \in \{alice, eve, bob\}$

$\cup \{charlie\}$

Solve each query more efficiently than starting from scratch for each query

Conference Contribution

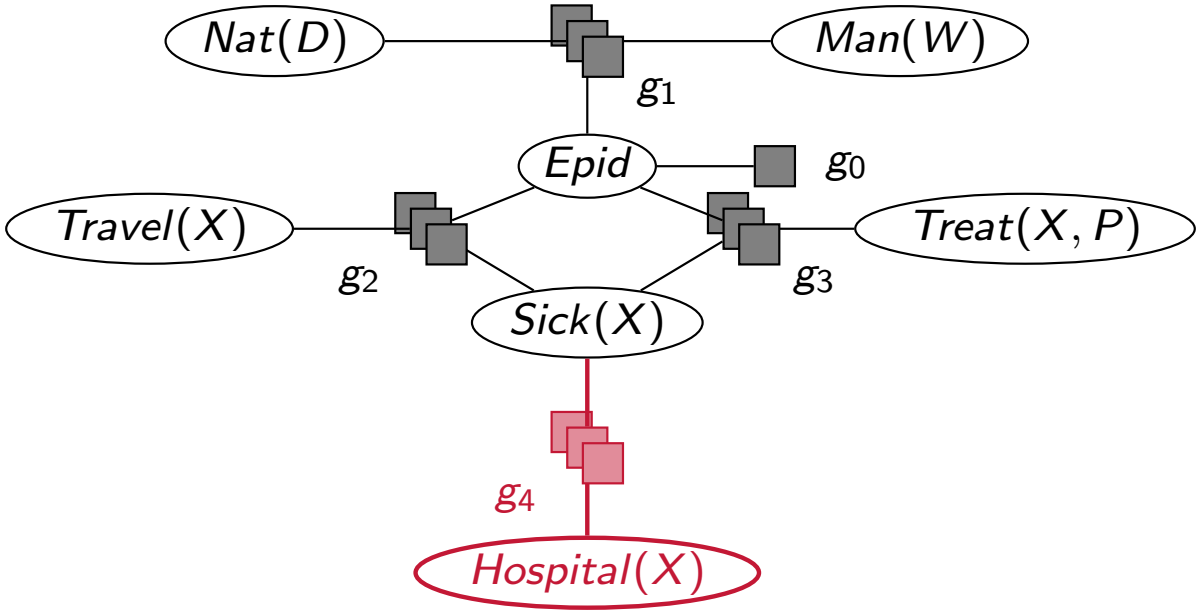
Adaptive Inference with LJT

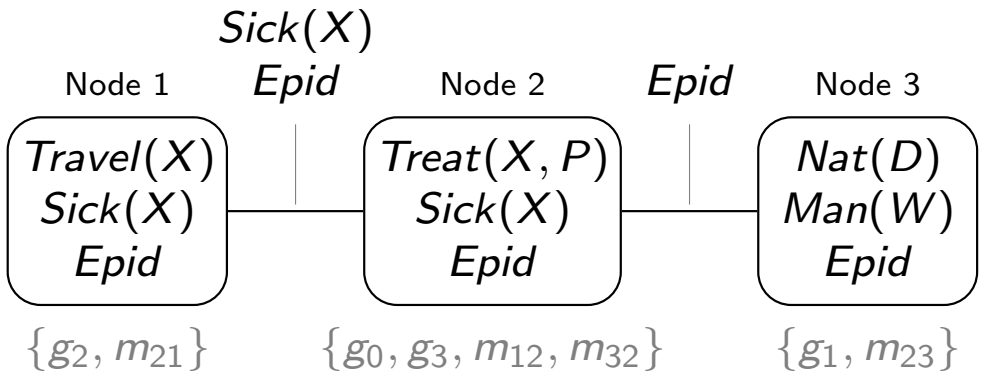
- Adjust a first-order junction tree to structural changes
  $\rightarrow$ While maintaining a valid tree
- Adapt local information to changes
  $\rightarrow$ Valid local information
- Adapt messages to changes
  $\rightarrow$ Valid outside information

Efficient, exact adaptive inference in probabilistic relational models

Adaptive Inference with LJT

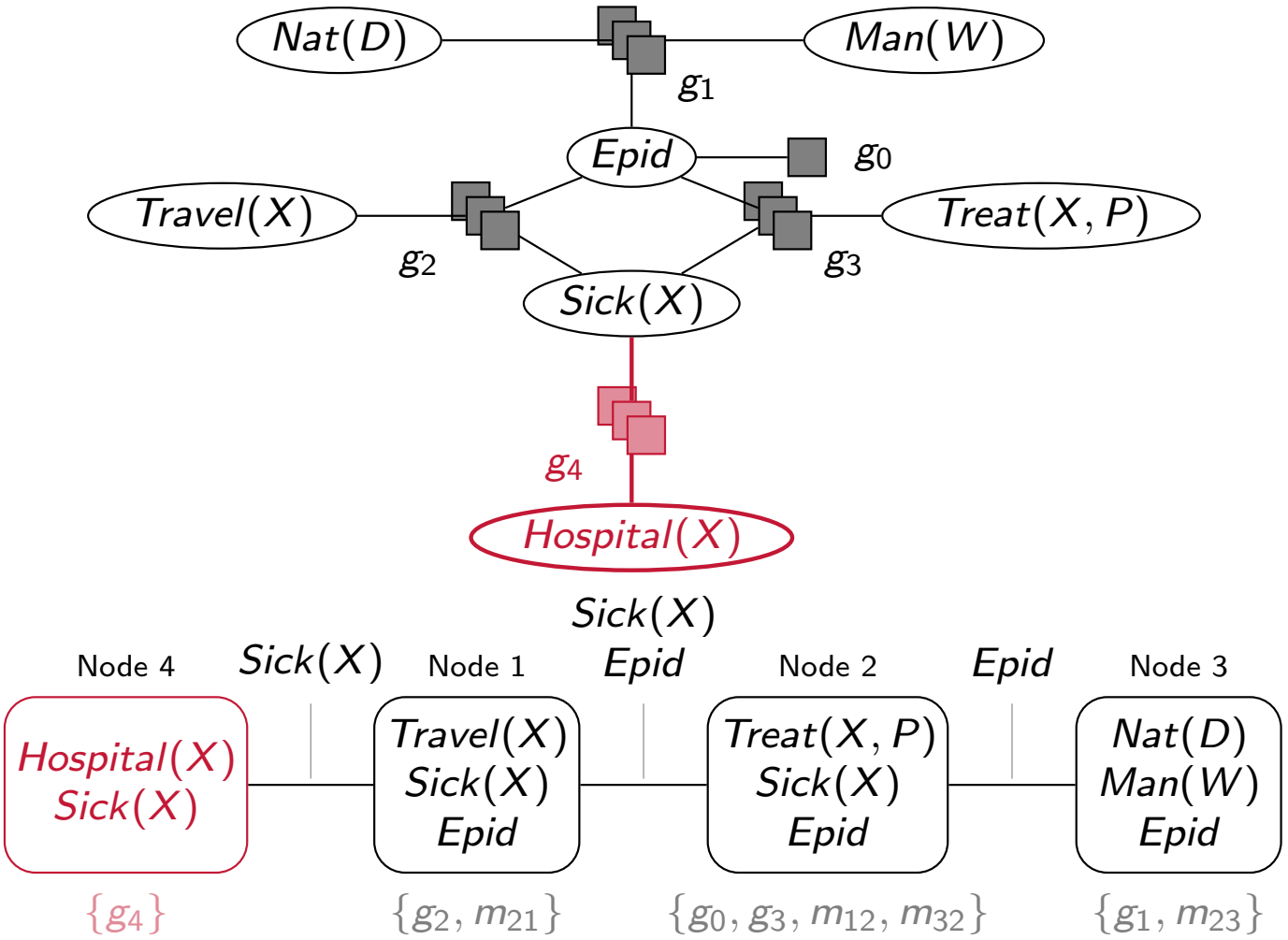
Structural Change: Adding a Parfactor with a New Variable





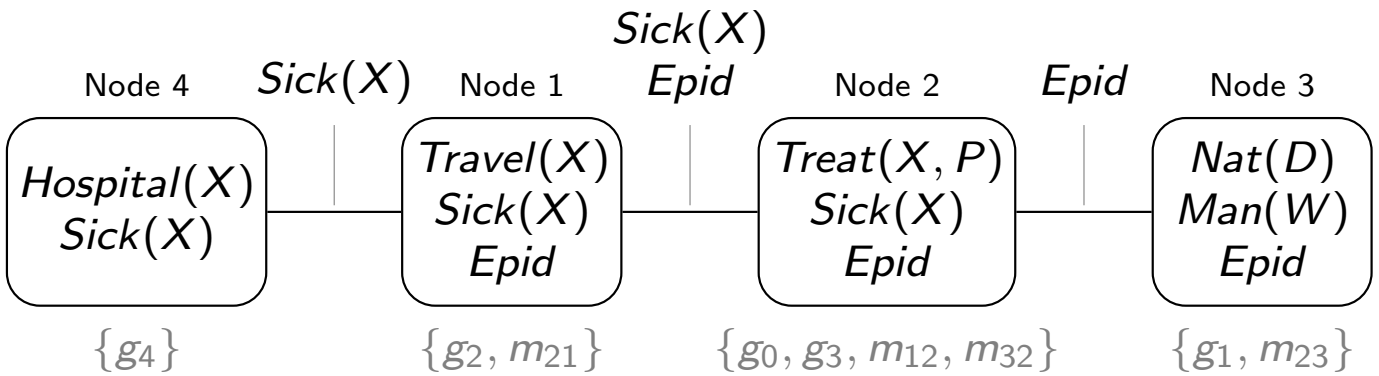
Adaptive Inference with LJT

Structural Change: Adding a Parfactor with a New Variable



Adaptive Inference with LJT

Adaptive Message Passing

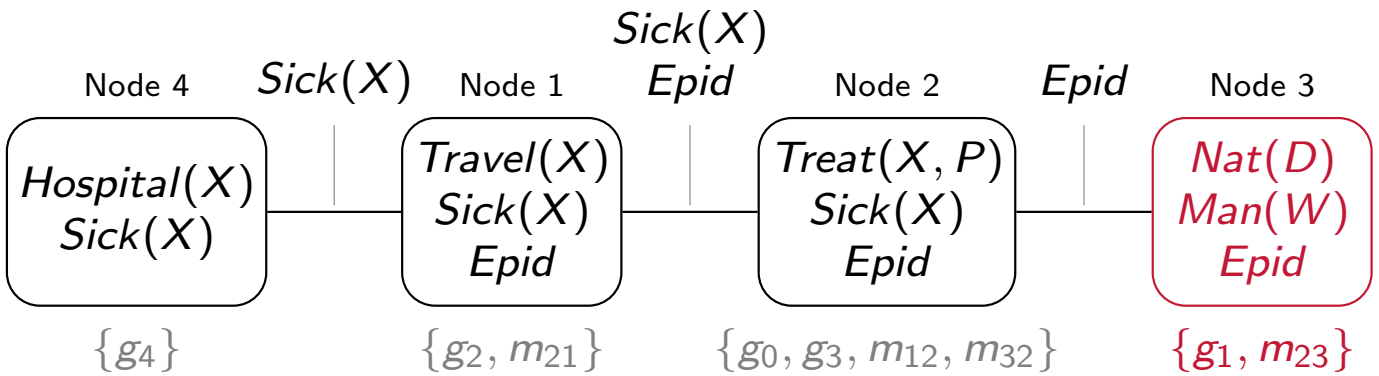


A node calculates a message if changes occur

- in its structure, local model, or in a message

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

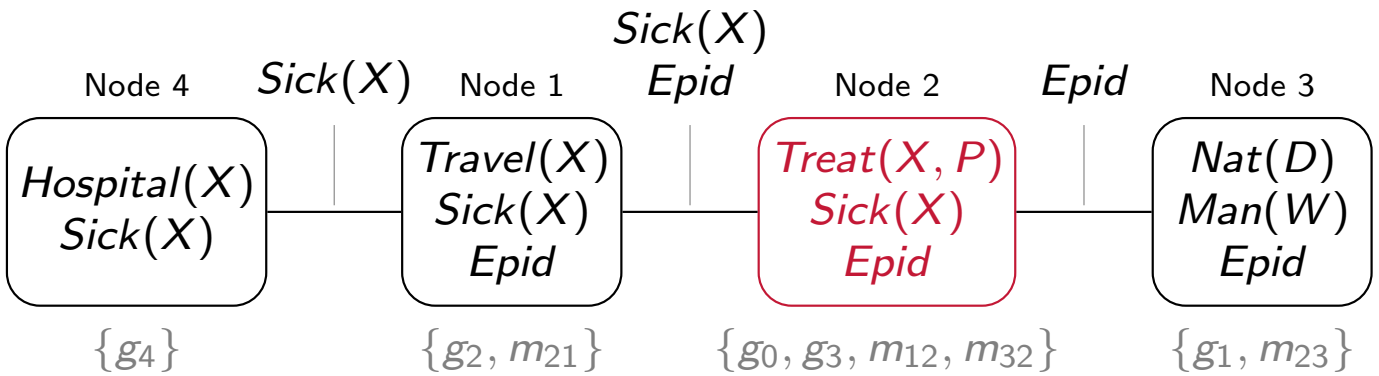
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

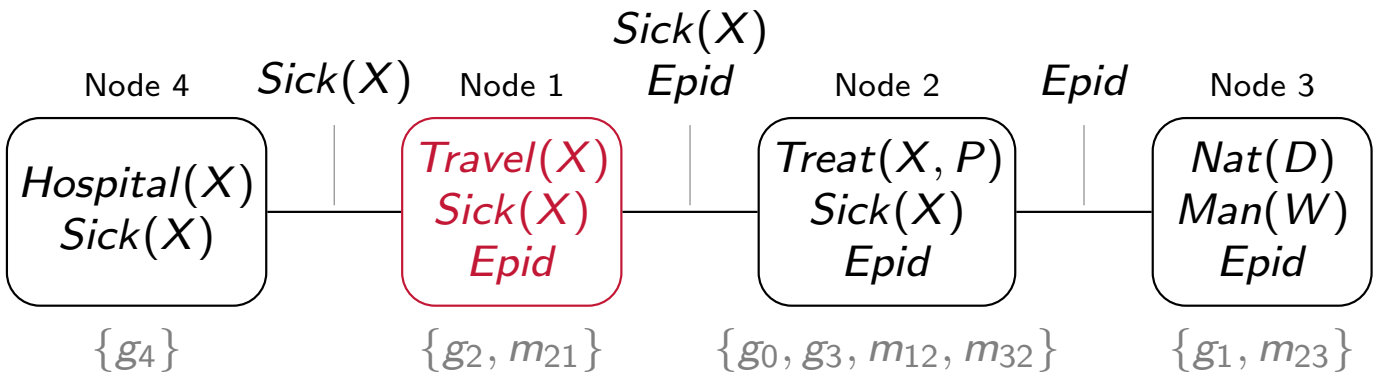
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

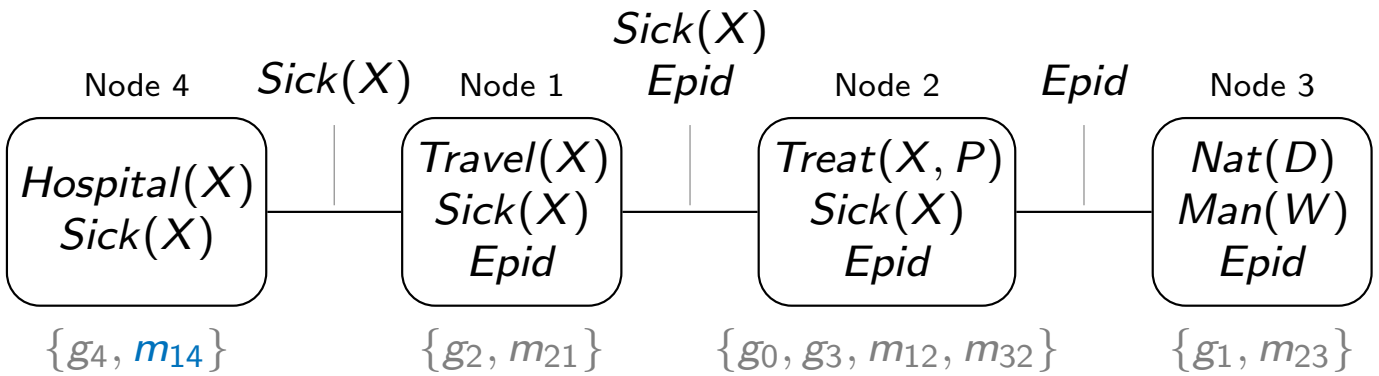
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

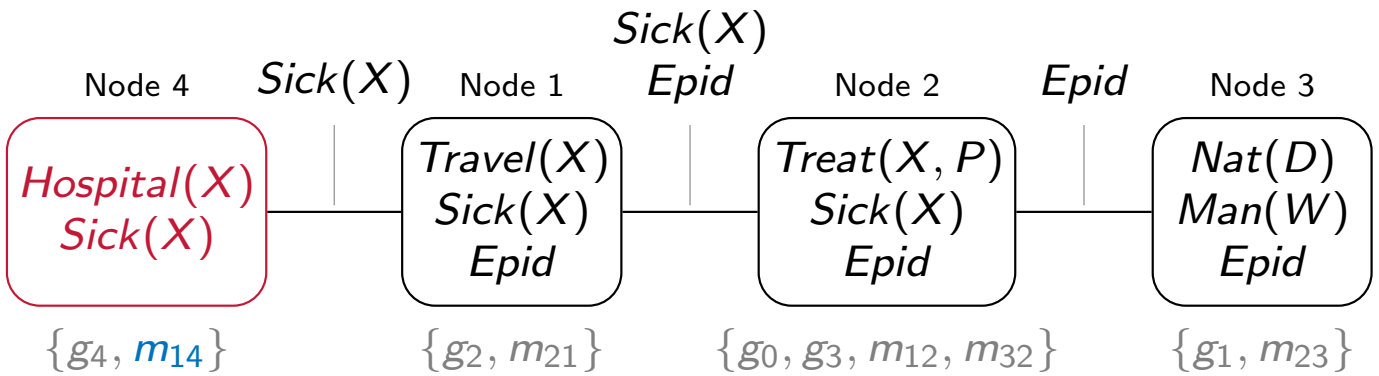
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

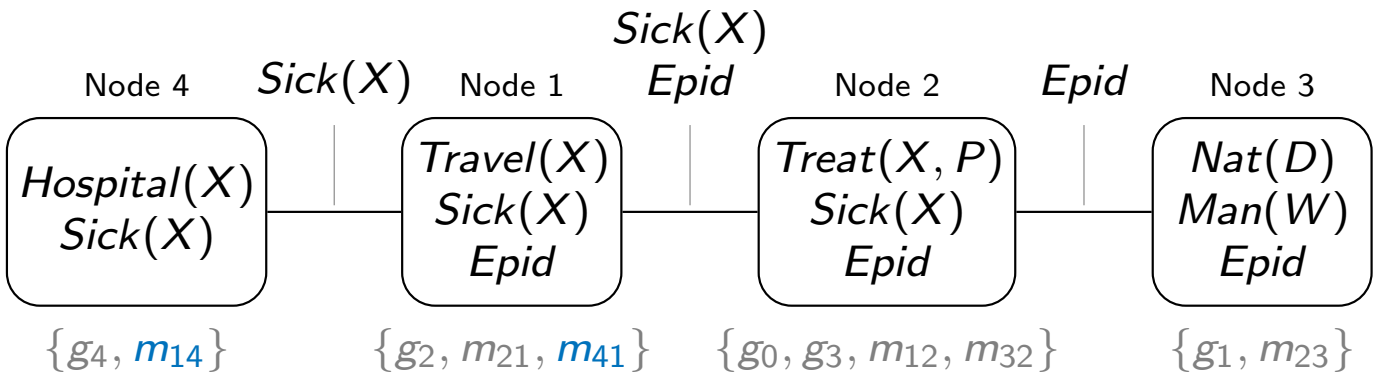
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour
- $4 \longrightarrow 1$: new node

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

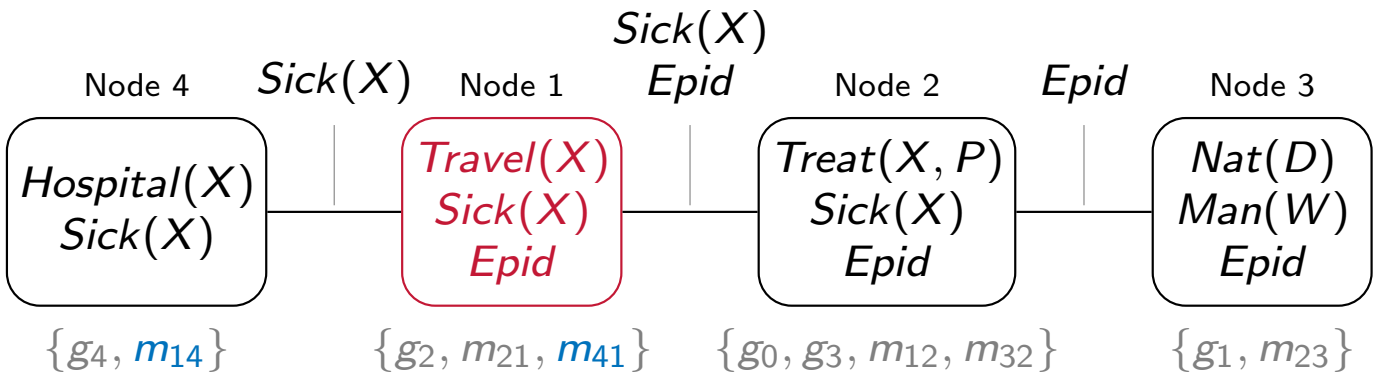
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour
- $4 \longrightarrow 1$: new node

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

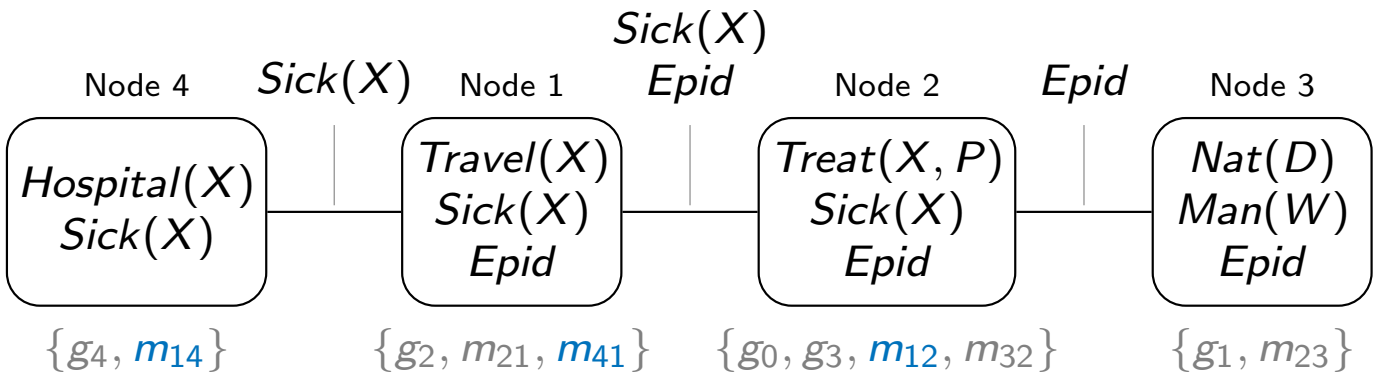
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour
- $4 \longrightarrow 1$: new node
- $1 \longrightarrow 2$: new message

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

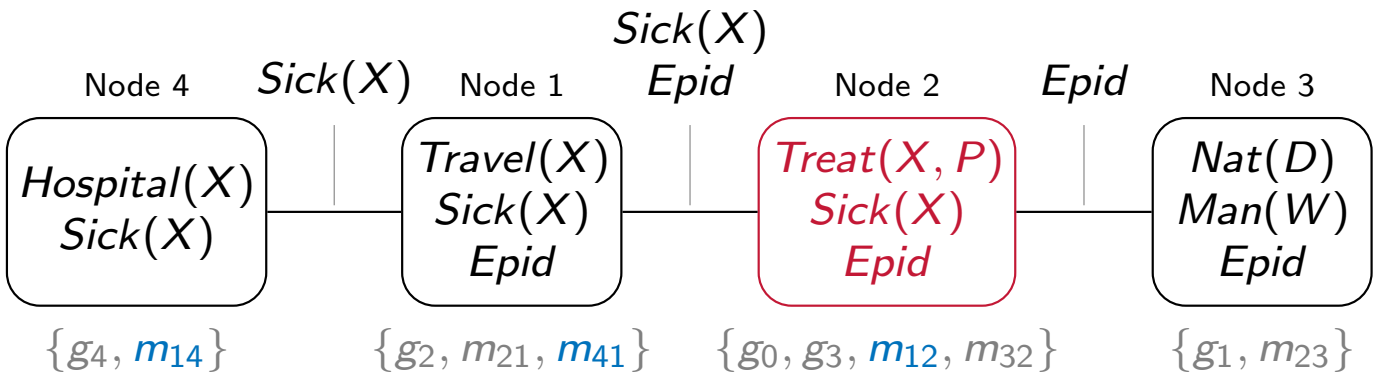
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour
- $4 \longrightarrow 1$: new node
- $1 \longrightarrow 2$: new message

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

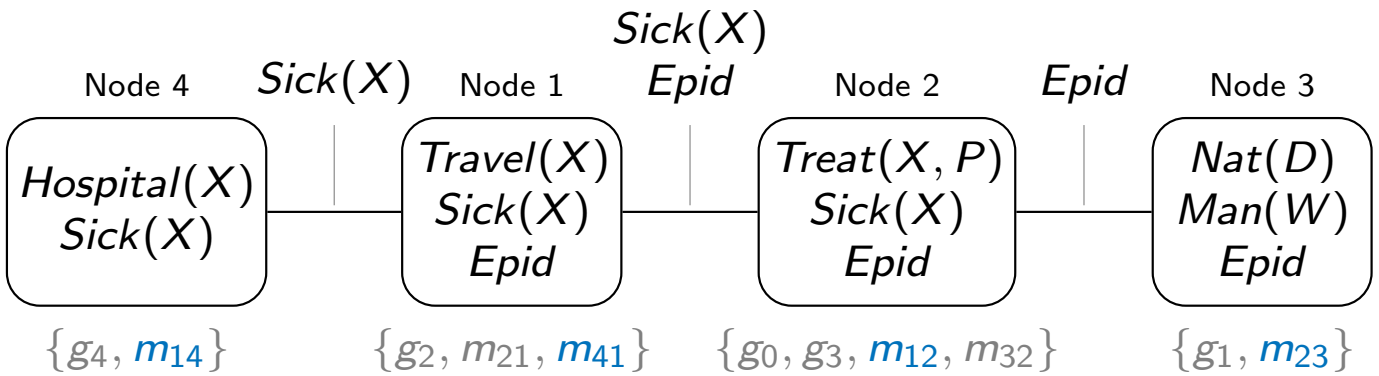
- in its structure, local model, or in a message

Messages

- 3 $\longrightarrow$ 2: ✓
- 2 $\longrightarrow$ 1: ✓
- 1 $\longrightarrow$ 4: new neighbour
- 4 $\longrightarrow$ 1: new node
- 1 $\longrightarrow$ 2: new message
- 2 $\longrightarrow$ 3: msg. changed

Adaptive Inference with LJT

Adaptive Message Passing



A node calculates a message if changes occur

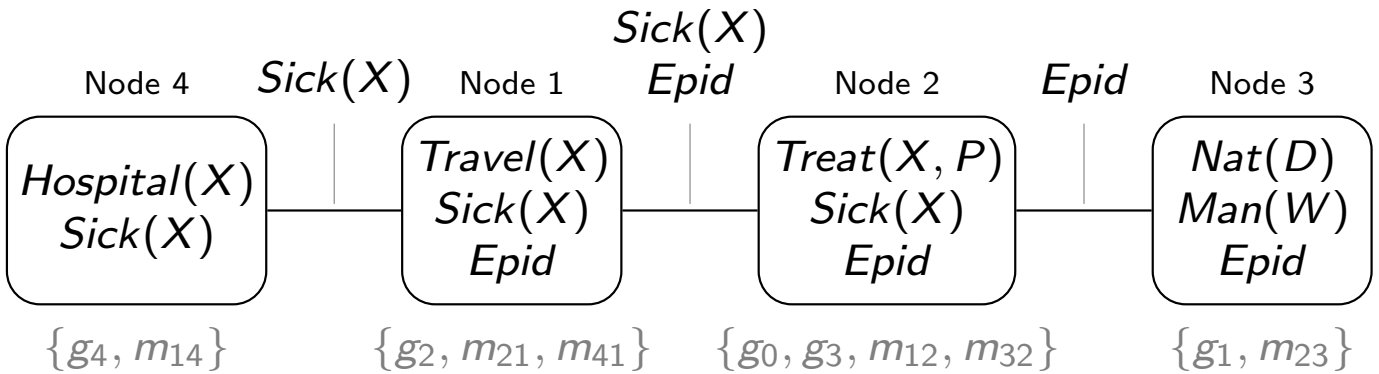
- in its structure, local model, or in a message

Messages

- $3 \longrightarrow 2$: ✓
- $2 \longrightarrow 1$: ✓
- $1 \longrightarrow 4$: new neighbour
- $4 \longrightarrow 1$: new node
- $1 \longrightarrow 2$: new message
- $2 \longrightarrow 3$: msg. changed

Adaptive Inference with LJT

Adaptive Evidence Entering

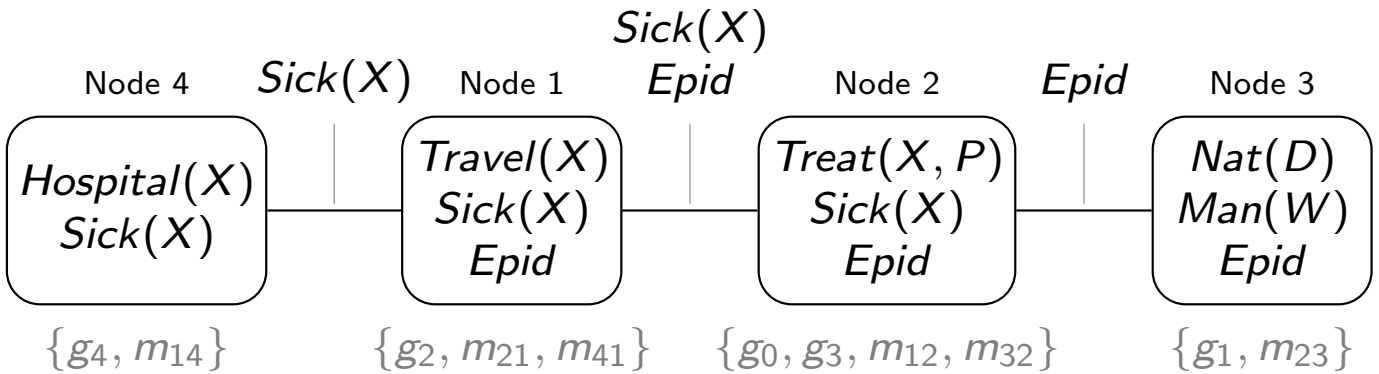


A node enters evidence if changes occur

- for its own variables

Adaptive Inference with LJT

Adaptive Evidence Entering



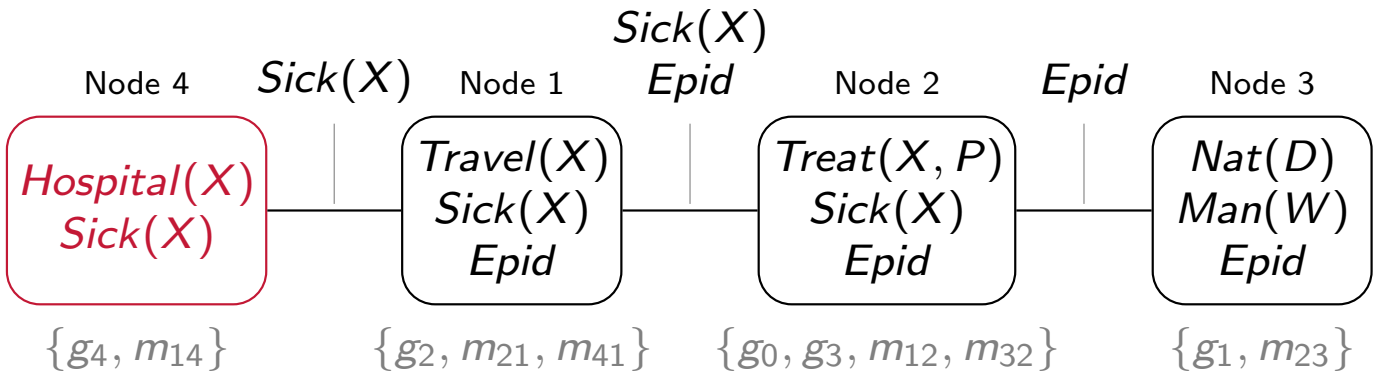
A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

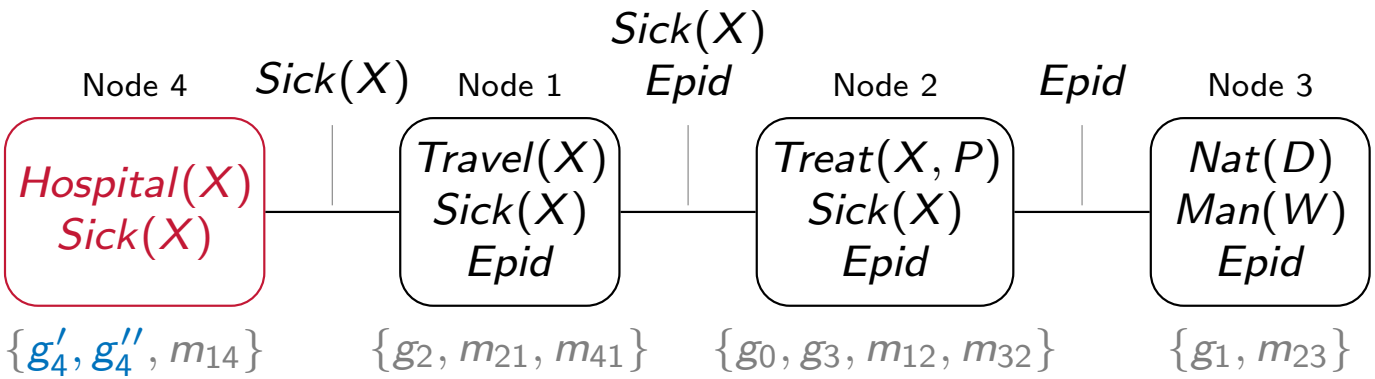
- for its own variables

Evidence: $Hospital(eve) = true$

- Node 4: enter evidence

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

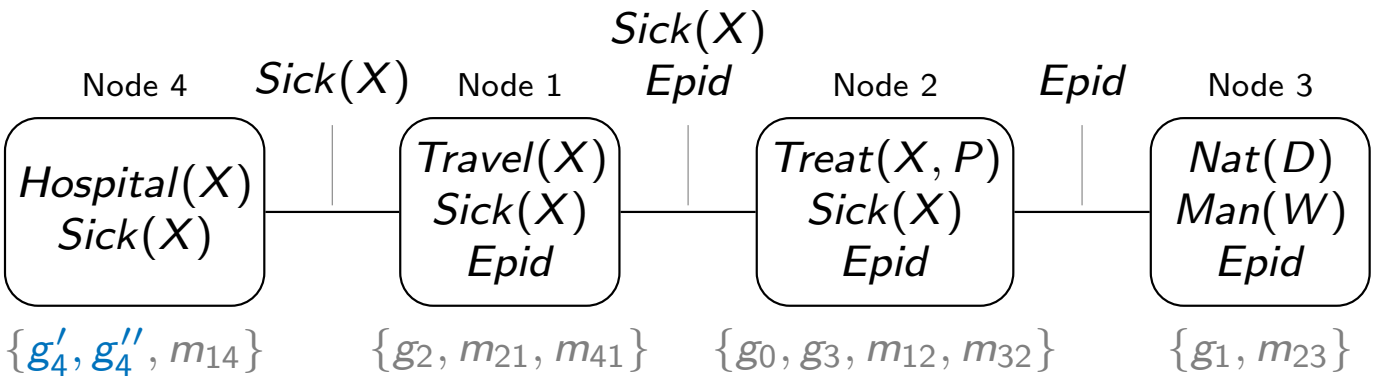
- for its own variables

Evidence: $Hospital(eve) = true$

- Node 4: enter evidence

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

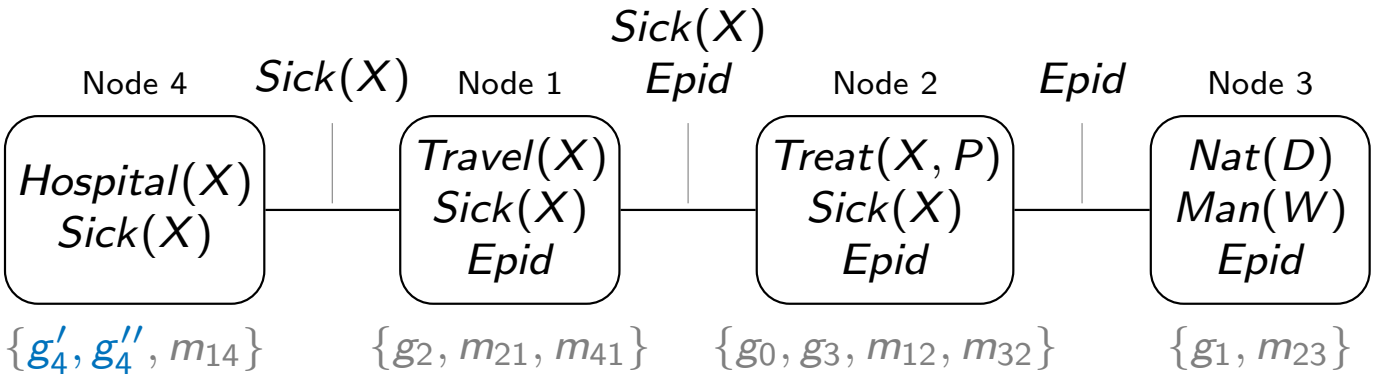
- for its own variables

Evidence: $Hospital(eve) = true$

- Node 4: enter evidence
- All other nodes: ✓

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

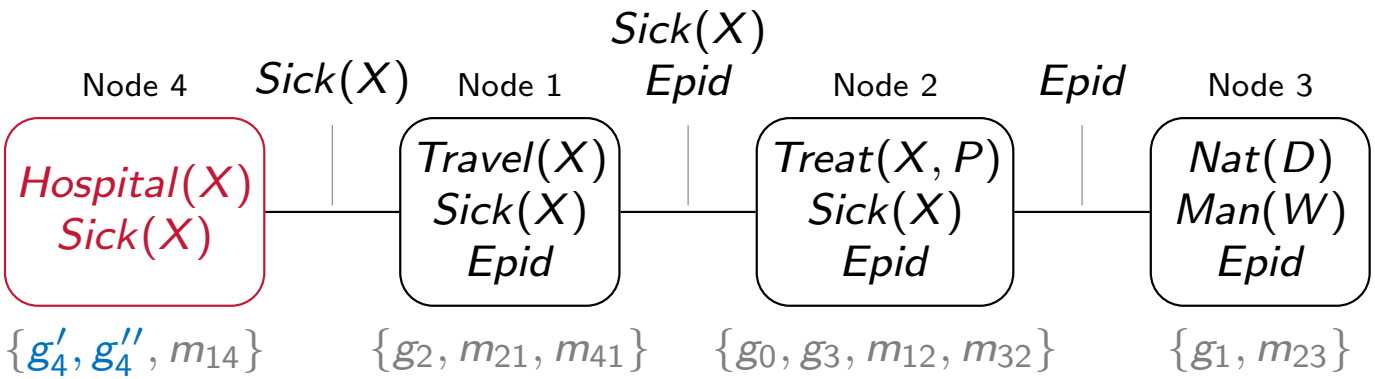
- for its own variables

Evidence: $Hospital(eve) = true$ Messages

- Node 4: enter evidence
- All other nodes: ✓

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

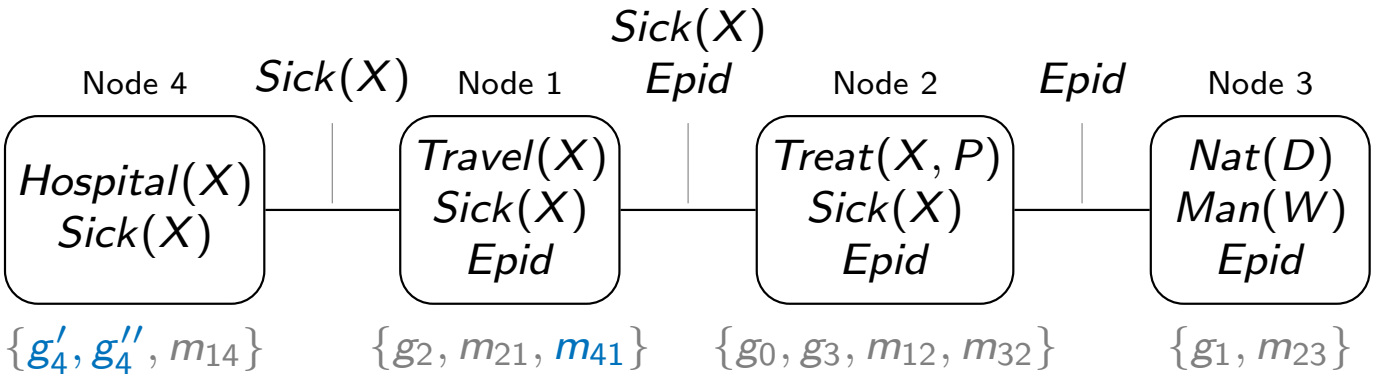
- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

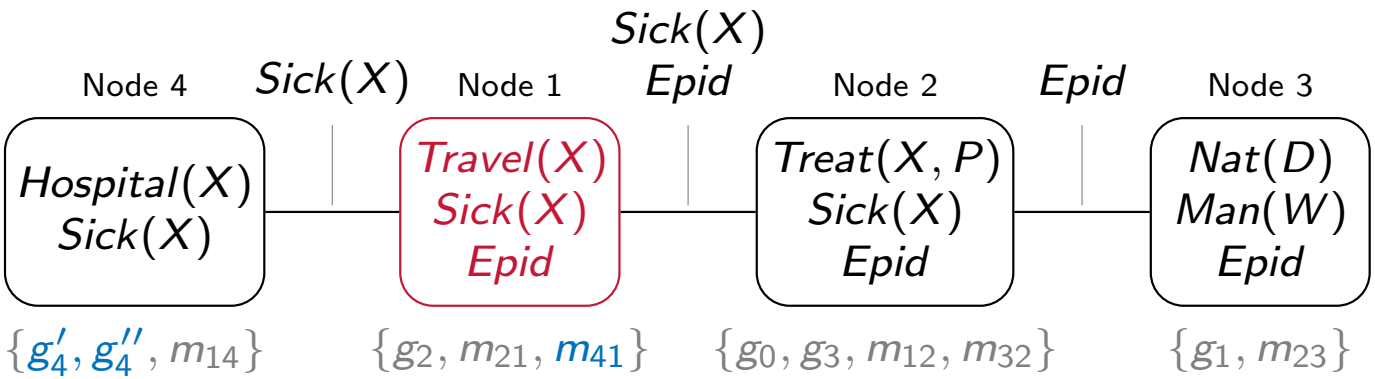
- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

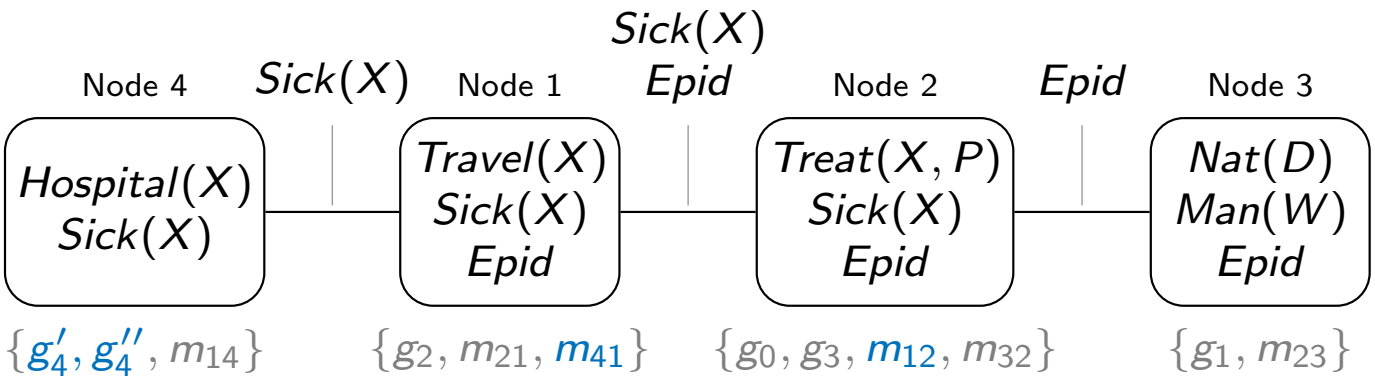
- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence
- $1 \longrightarrow 2$: msg. changed

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

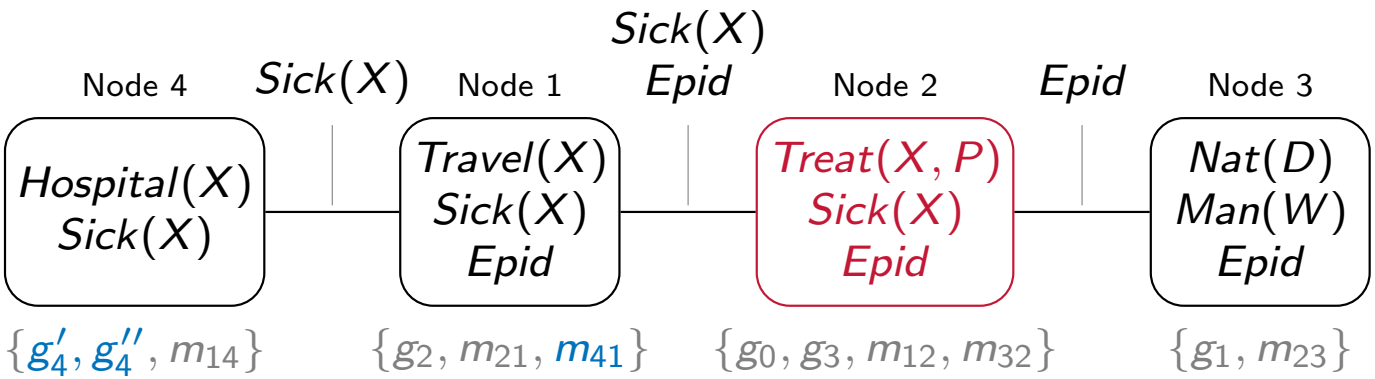
- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence
- $1 \longrightarrow 2$: msg. changed

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

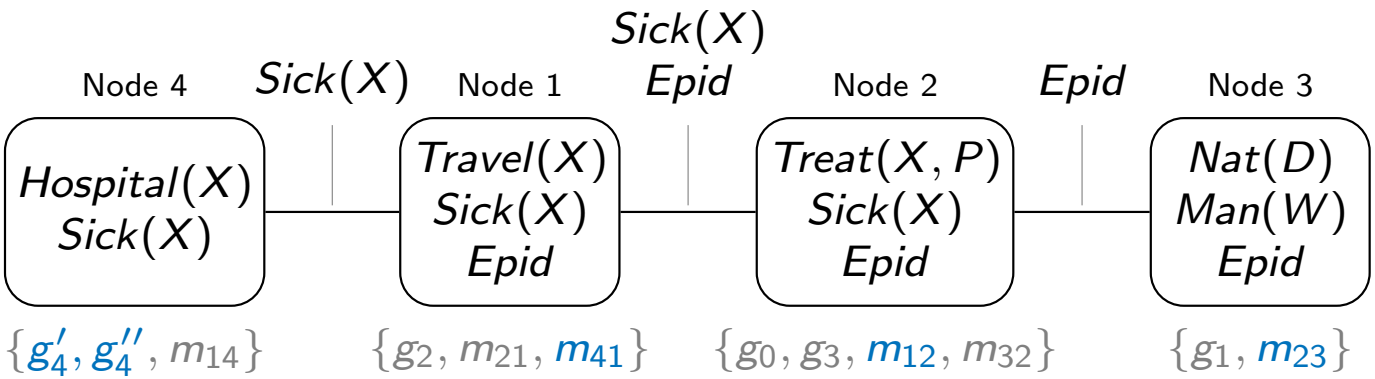
- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence
- $1 \longrightarrow 2$: msg. changed
- $2 \longrightarrow 3$: msg. changed

Adaptive Inference with LJT

Adaptive Evidence Entering



A node enters evidence if changes occur

- for its own variables

Evidence: $Hospital(eve) = true$

- Node 4: enter evidence
- All other nodes: ✓

Messages

- $4 \longrightarrow 1$: new evidence
- $1 \longrightarrow 2$: msg. changed
- $2 \longrightarrow 3$: msg. changed

Test Run

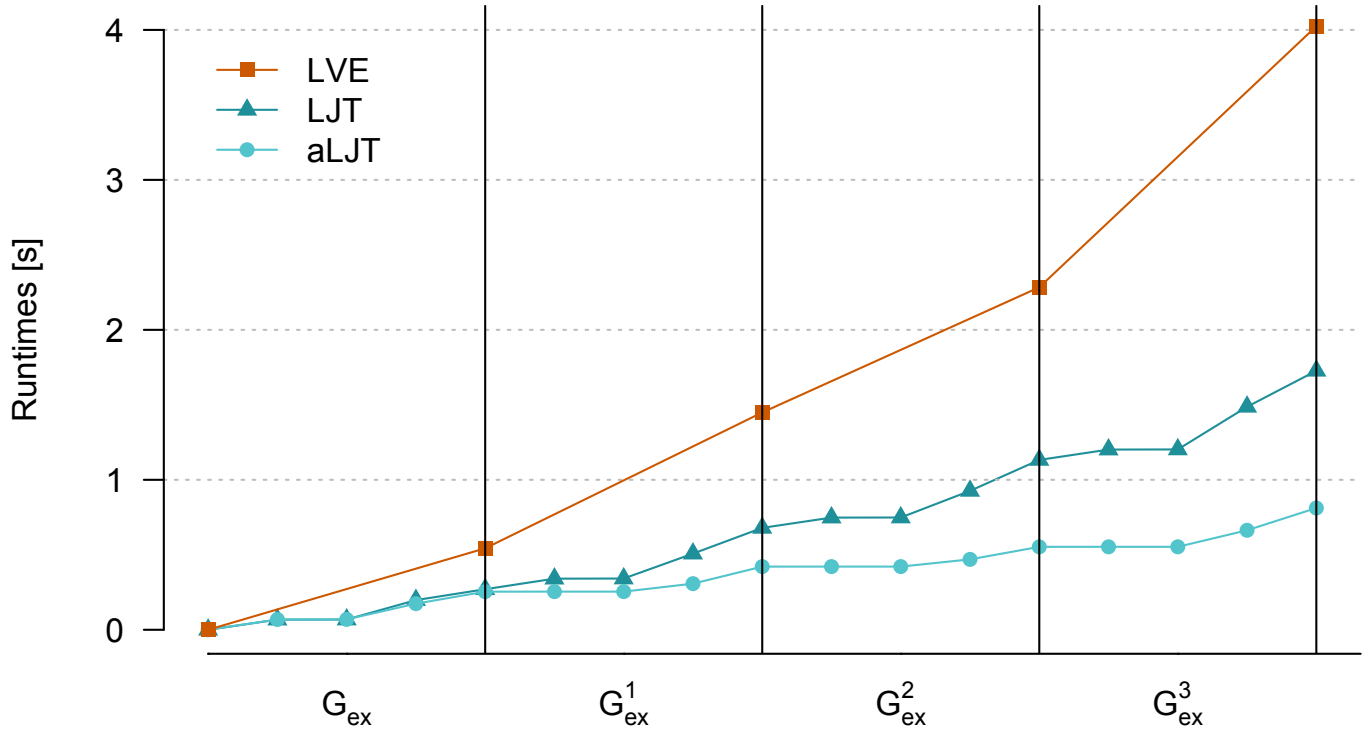


Figure: Runtimes [ms] for two queries and three changes (points connected for readability)

`lve`: Implementation by Taghipour (2013)

`(a)ljt`: Our implementation of (adaptive) LJT

Analysis: Adaptive Inference with LJT

Algorithm steps

1. Adaptation of tree to changes
2. Adaptive evidence entering
3. Adaptive message passing
4. Query answering

Ensure correct information

$\rightarrow$ Redo what is necessary

Adaptive steps

$\rightarrow$ Save up to 50% of message passing runtime (most expensive step)

Changes too drastic/too many

$\rightarrow$ Reconstruct tree from scratch

Conference Contribution

Adaptive Inference with LJT

- Adapt a first-order junction tree to structural changes
  $\rightarrow$ Valid representation of model
- Adapt local information to changes
  $\rightarrow$ Valid local information
- Adapt messages to changes
  $\rightarrow$ Valid overall information

Efficient, exact adaptive inference in probabilistic relational models

Conference Contribution

Adaptive Inference with LJT

- Adapt a first-order junction tree to structural changes
  $\rightarrow$ Valid representation of model
- Adapt local information to changes
  $\rightarrow$ Valid local information
- Adapt messages to changes
  $\rightarrow$ Valid overall information

Efficient, exact adaptive inference in probabilistic relational models

Future work

- Learning lifted representations
- Modelling decision making