Exact Inference

Kayhan Batmanghelich

Probabilistic Inference and Learning

- We now have compact representations of probability distributions: **Graphical Models**
- A GM *M* describes a unique probability distribution *P*

- Typical tasks:
  - Task 1: How do we answer **queries** about P_M, *e.g.*, $P_M(X|Y)$?
    - We use **inference** as a name for the process of computing answers to such queries
  - Task 2: How do we estimate a **plausible model** *M* from data *D*?
    1. We use **learning** as a name for the process of obtaining point estimate of *M*.
    2. But for *Bayesian*, they seek $p(M \mid D)$, which is actually an **inference** problem.
    3. When not all variables are observable, even computing point estimate of *M* need to do **inference** to impute the *missing data*.

By this I mean the graph structure and/or the parameters

Query 1: Likelihood

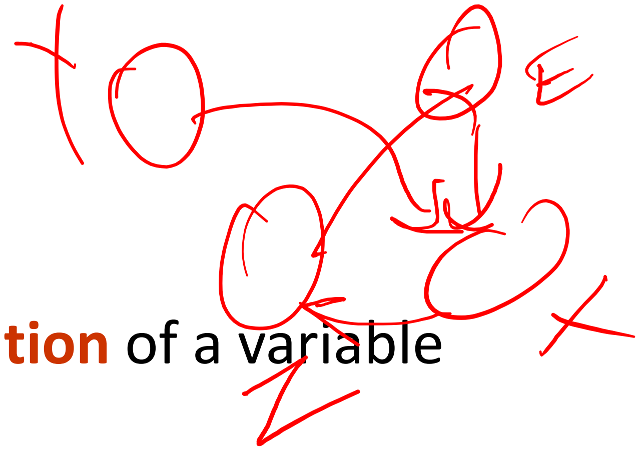
- Most of the queries one may ask involve **evidence**
  - Evidence $\mathbf{e}$ is an assignment of values to a set $\mathbf{E}$ variables in the domain
  - Without loss of generality $\mathbf{E} = \{ X_{k+1}, \ldots, X_n \}$

- Simplest query: compute probability of evidence

$$P(\mathbf{e}) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, \ldots, x_k, \mathbf{e})$$

  - this is often referred to as computing the **likelihood** of $\mathbf{e}$

Query 2: Conditional Probability



- Often we are interested in the **conditional probability distribution** of a variable given the evidence

$$P(X \mid \mathbf{e}) = \frac{P(X,\mathbf{e})}{P(\mathbf{e})} = \frac{P(X,\mathbf{e})}{\sum_x P(X = x,\mathbf{e})}$$

evidence

x, e, y

P(Y|e)

  - this is the ***a posteriori* belief** in X, given evidence $\mathbf{e}$

- **Marginalization:** the process of summing out the “unobserved” or "don't care" variables z is called.

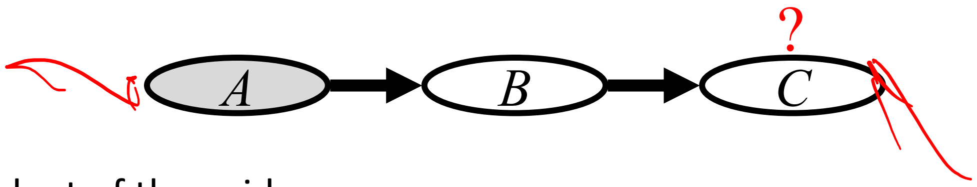
$$P(\mathbf{Y} \mid e) = \sum_{\mathbf{z}} P(\mathbf{Y},\mathbf{Z} = \mathbf{z} \mid \mathbf{e})$$

e, x, y

$P(x_1|x_2)$

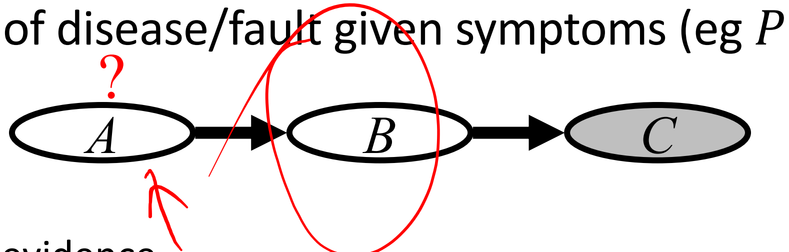
Applications of a posteriori Belief

- **Prediction** : what is the probability of an outcome given the starting condition (eg $P(C|A)$)



  - the query node is a descendent of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms (eg $P(A|C)$)



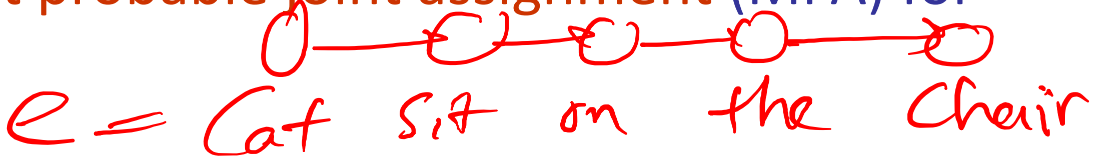
  - the query node an ancestor of the evidence

- ***Note***: The directionality of information flow between variables is not restricted by the directionality of the edges in a GM.
- You will see more application during **Learning** under partial observation

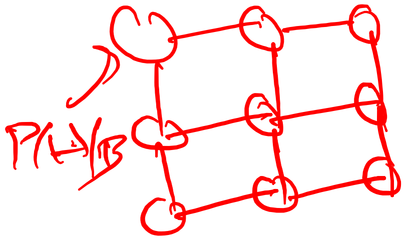
$\max_{x_1, \dots x_4} P(x_1, \dots x_4 | e)$

Query 3: Most Probable Assignment

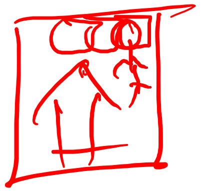
- In this query we want to find the most probable joint assignment (MPA) for *some* variables of interest



- Such reasoning is usually performed under some given evidence e, and ignoring (the values of) other variables z :

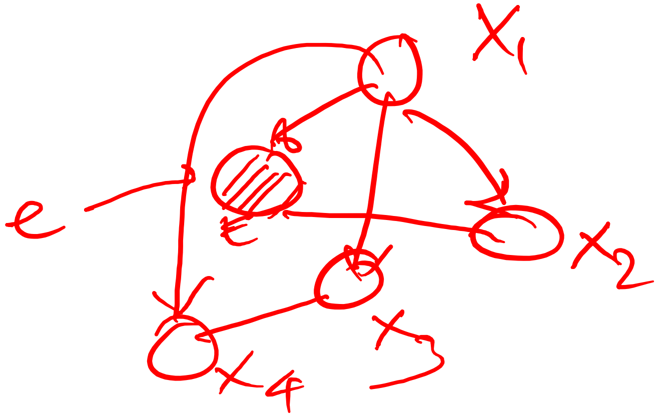


$$\mathrm{MPA}(\mathbf{Y}|\mathbf{e}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{e}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{\mathbf{z}} P(\mathbf{y},\mathbf{z}|\mathbf{e})$$



  - this is the maximum *a posteriori* configuration of y.

$P(X_1, X_2, X_3, X_4 | E)$



Applications of MPA

- Classification
  - find most likely label, given the evidence
- Explanation
  - what is the most likely scenario, given the evidence

Cautionary note:

- The MPA of a variable depends on its "context"---the set of variables been jointly queried
- Example:
  - MPA of Y_1 ?
  - MPA of (Y_1, Y_2) ?

max $P(Y_1)$

$$\sum_{Y_2} P(Y_1, Y_2)$$

y_1	y_2	$P(y_1, y_2)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

0.4

0.6

Complexity of Inference

Thm:

Computing $P(\mathrm{X}=\mathrm{x}\mid\mathrm{e})$ in a GM is NP-hard

- Hardness does not mean we cannot solve inference. It simply says that there exist difficult inference problems. It depends on the structure.
  - It implies that we cannot find a general procedure that works efficiently for arbitrary GMs
  - For particular families of GMs, we can have provably efficient procedures

Landscape of inference algorithms

- **Exact inference algorithms**
  - The elimination algorithm
  - Message-passing algorithm (sum-product, belief propagation)
  - The junction tree algorithms

Classical

- **Approximate inference techniques**
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
  - Variational algorithms

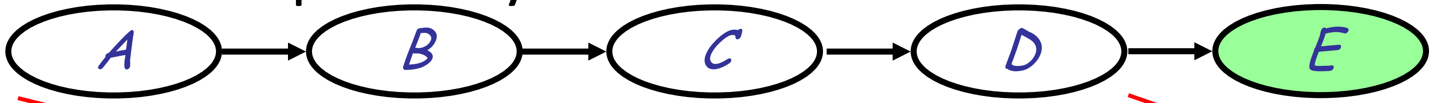
Variable Elimination

Marginalization and Elimination

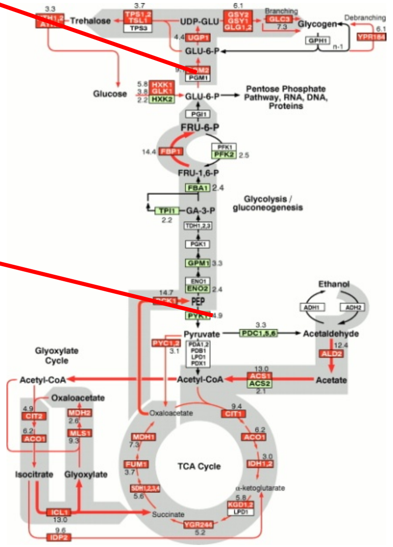
$E = \{e_1, \dots, e_k\}$

$E = e_1$

- A signal transduction pathway:



What is the likelihood that protein E is active?



4

- Query: $P(\boldsymbol{e})$

$$P(e) = \underbrace{\sum_d \sum_c \sum_b \sum_a}_{} P(a,b,c,d,e)$$

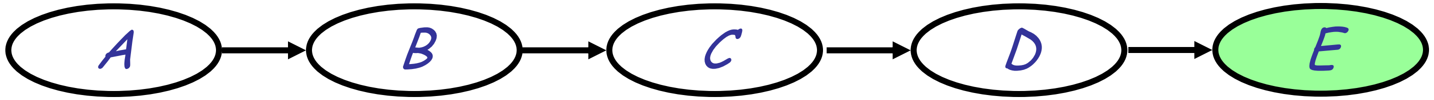
K^4

a naïve summation needs to enumerate over an exponential number of terms

- By chain decomposition, we get

$$= \sum_d \sum_c \sum_b \sum_a P(a)P(b \mid a)P(c \mid b)P(d \mid c)P(e \mid d)$$

Elimination on Chains



$O(nK^2)$

$O(K^2)$

- Rearranging terms (First sum a, then b, ...)

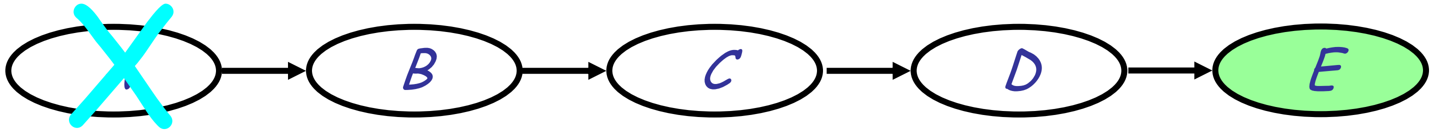
$$P(e)=\sum_d\sum_c\sum_b\boxed{\sum_a P(a)P(b\,|\,a)}P(c\,|\,b)P(d\,|\,c)P(e\,|\,d)$$

$$=\sum_d\sum_c\sum_b P(c\,|\,b)P(d\,|\,c)P(e\,|\,d)\underbrace{\sum_a P(a)P(b\,|\,a)}_{m(b)}$$

$v\in\Delta^K$

$\underbrace{K\left[\quad\right]}_{K}\Delta$

Elimination on Chains



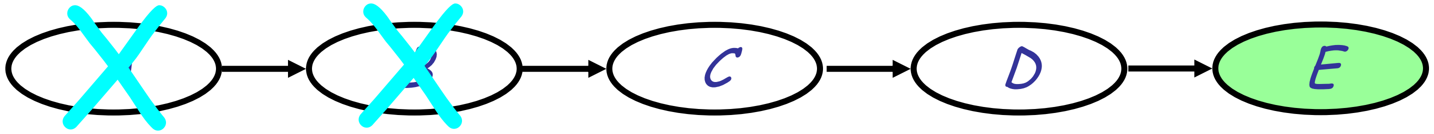
- Now we can perform innermost summation

$$P(e) = \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) \sum_a P(a) P(b \mid a)$$

$$= \sum_d \sum_c \boxed{\sum_b P(c \mid b)} P(d \mid c) P(e \mid d) p(b)$$

- This summation "eliminates" one variable from our summation argument at a "local cost".

Elimination in Chains

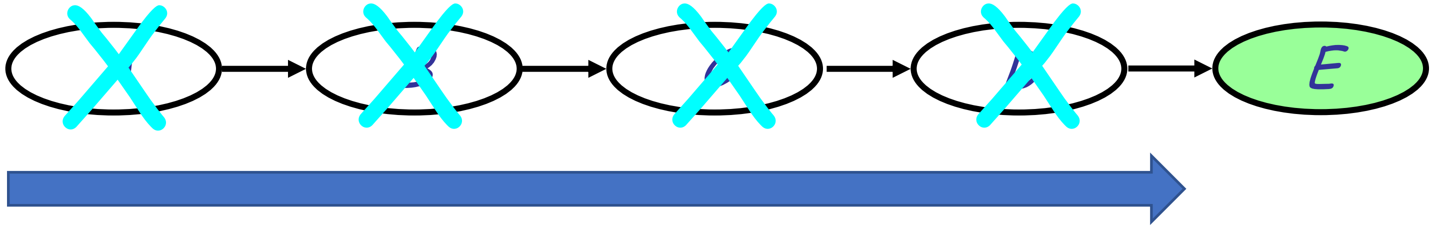


- Rearranging and then summing again, we get

$$
\begin{aligned}
P(e) &= \sum_d \sum_c \sum_b P(c \mid b) P(d \mid c) P(e \mid d) p(b) \\
&= \sum_d \sum_c P(d \mid c) P(e \mid d) \boxed{\sum_b P(c \mid b) p(b)} \\
&= \sum_d \sum_c P(d \mid c) P(e \mid d) \boxed{p(c)}
\end{aligned}
$$

$A(M_A(M_B(M_C M_D)))\dots$

Elimination in Chains

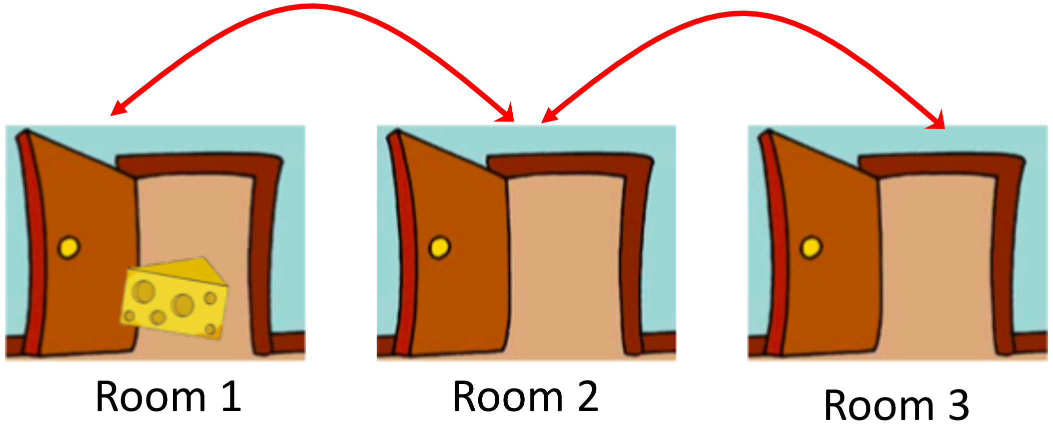


- Eliminate nodes one by one all the way to the end, we get

$$P(e) = \sum_d P(e \mid d) p(d)$$

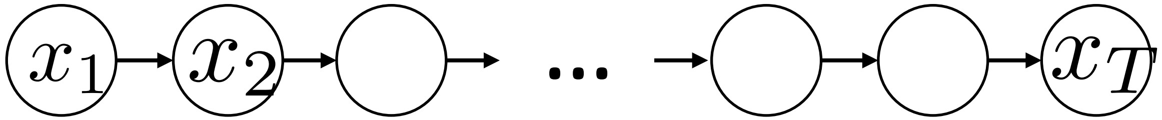
- Complexity:
  - A clever elimination takes $O(nk^2)$ *versus* $O(k^n)$ for the naïve approach.

Associativity of matrix multiplication



$$p(x_{t+1} = i | x_t = j) = M_{ij}$$

$$\begin{bmatrix} 0.7 & 0.5 & 0 \\ 0.3 & 0.3 & 0.5 \\ 0 & 0.2 & 0.5 \end{bmatrix}$$



$$p(x_1, \cdots, x_T) = p(x_1) \prod_{t=1}^{T-1} p(x_{t+1} | x_t)$$

$P \quad = P(x_2|x_1)P(x_1)$

Associativity of matrix multiplication

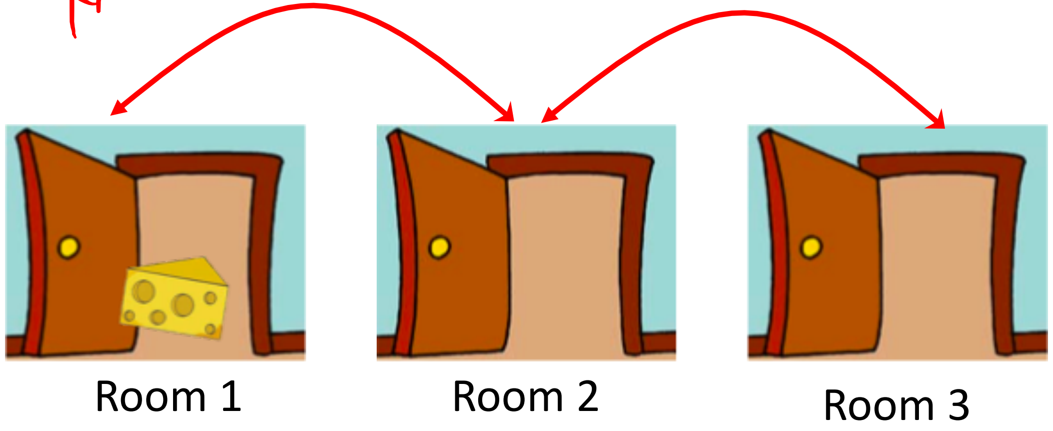
$P(x) = v$

$v \in \mathbb{R}^3$

$Mv = \lambda v$

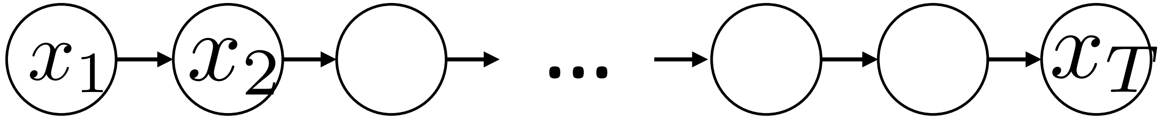
$\cancel{v}$

$MP = P$



$$p(x_{t+1} = i | x_t = j) = M_{ij}$$

$$\begin{bmatrix} 0.7 & 0.5 & 0 \\ 0.3 & 0.3 & 0.5 \\ 0 & 0.2 & 0.5 \end{bmatrix}$$



$$p(x_5 = i | x_1 = 1) = \sum_{x_4, x_3, x_2} p(x_5|x_4)p(x_4|x_3)p(x_3|x_2)p(x_2|x_1 = 1)$$

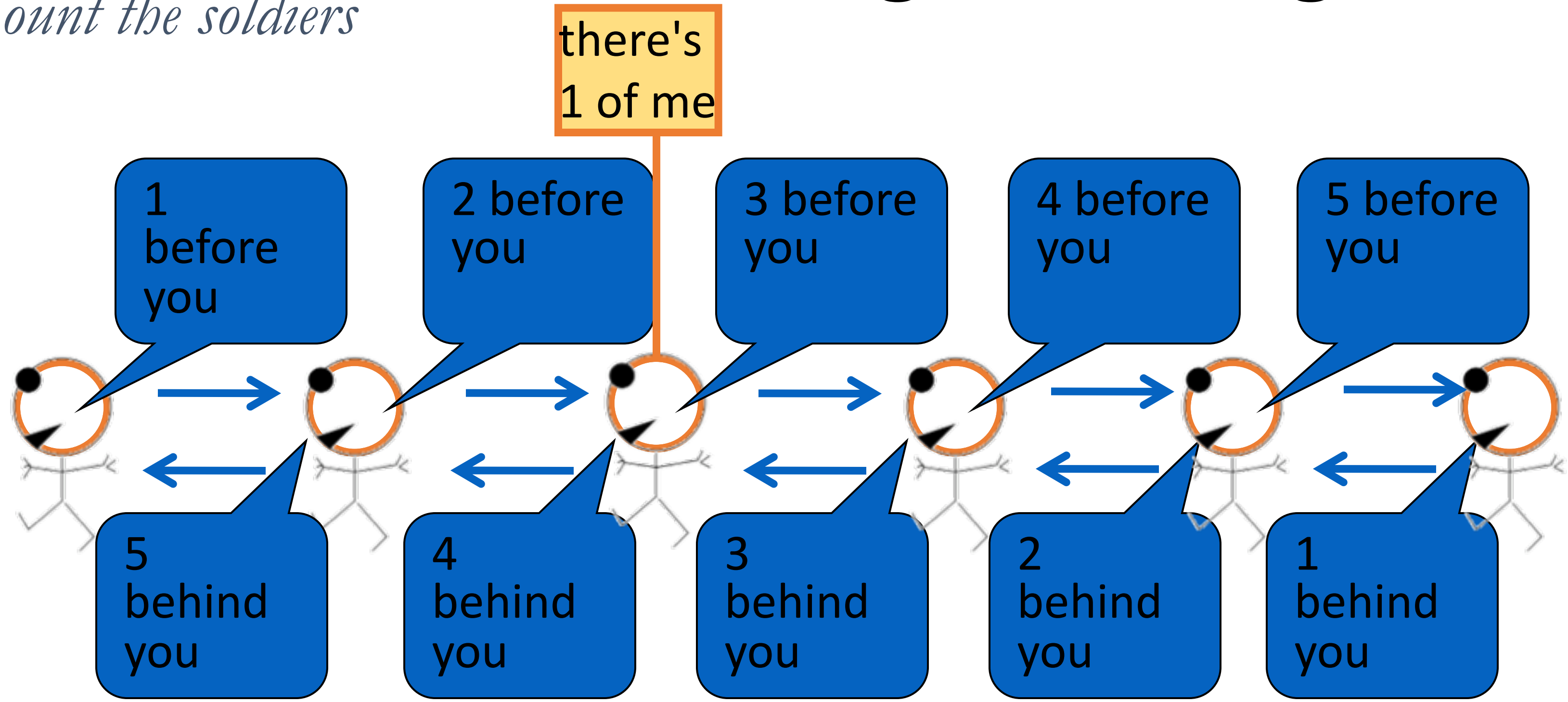
$$= [M^4 v]_i$$

$MP_\infty = P_\infty$

$T \to \infty$

Great Ideas in ML: Message Passing

Count the soldiers



there's 1 of me

1 before you

2 before you

3 before you

4 before you

5 before you

5 behind you

4 behind you

3 behind you

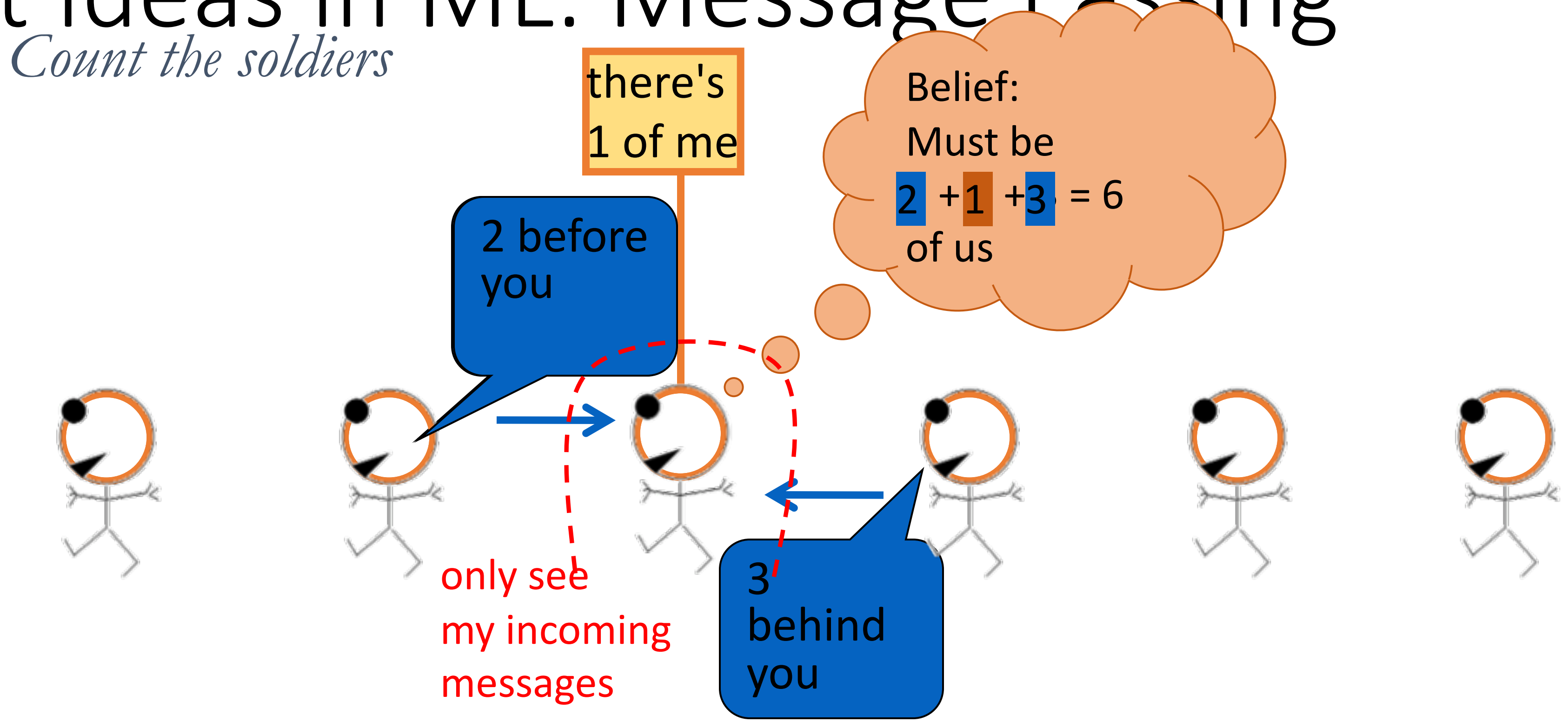
2 behind you

1 behind you

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Count the soldiers



there's 1 of me

2 before you

Belief: Must be 2 + 1 + 3 = 6 of us

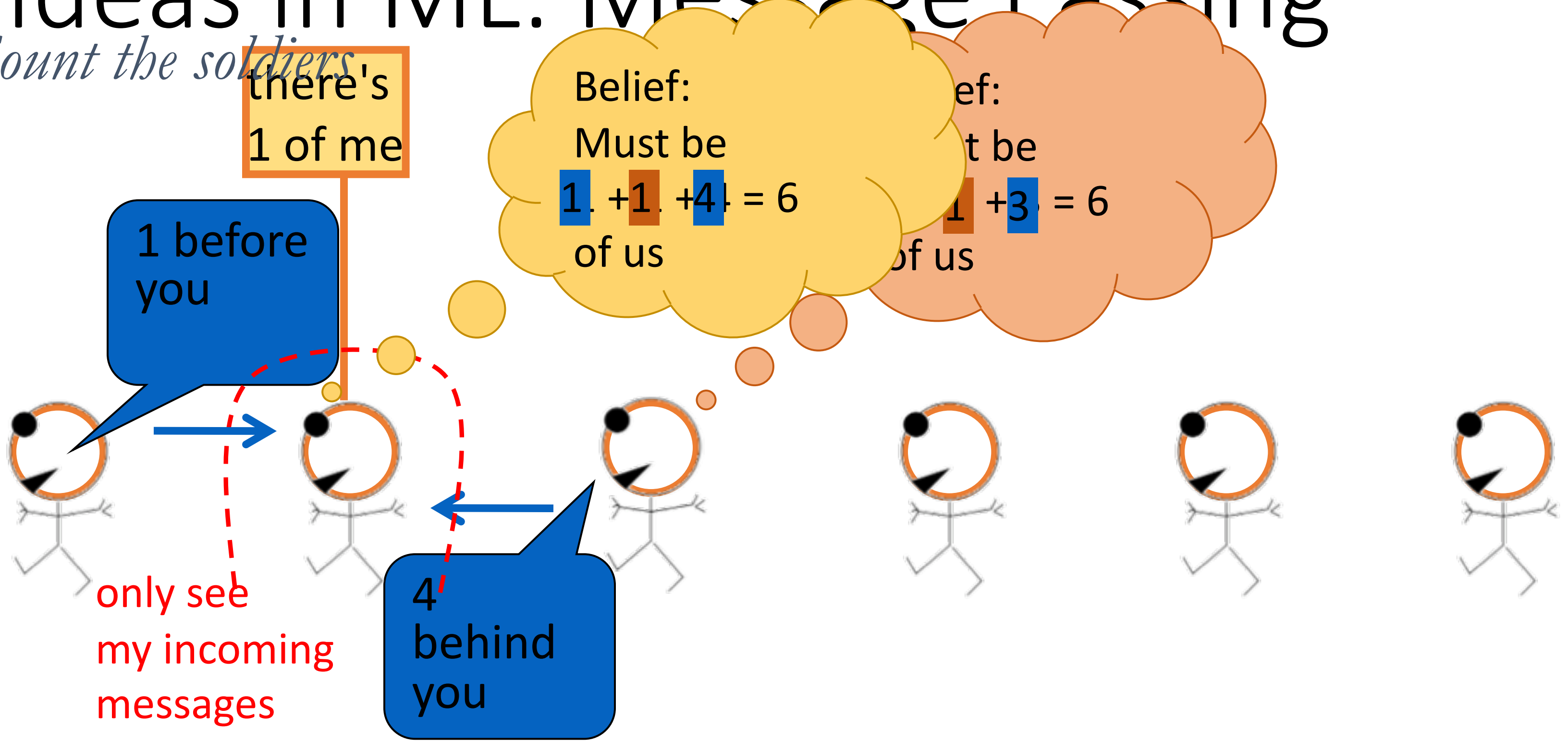
only see my incoming messages

3 behind you

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Count the soldiers

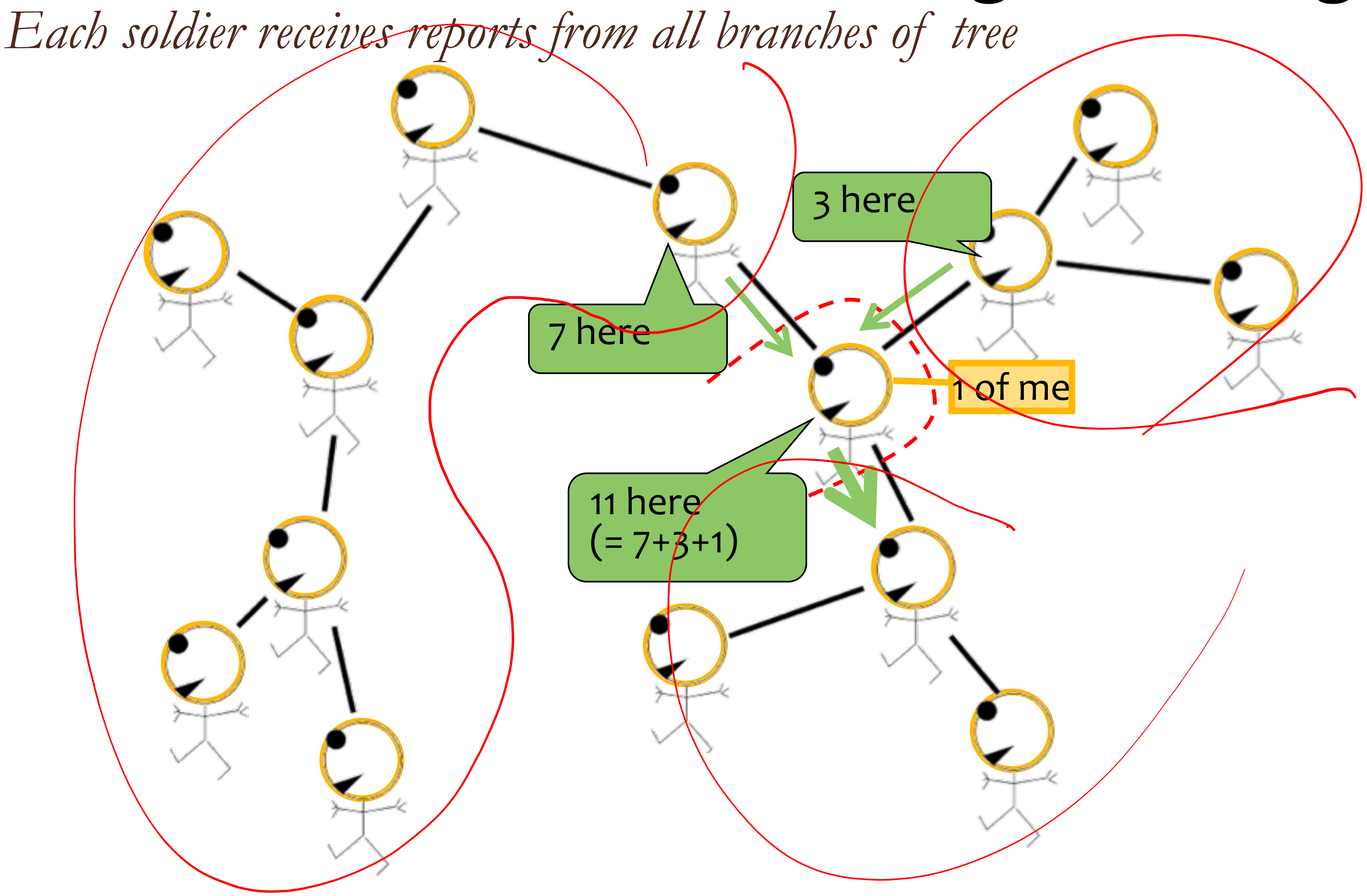

there's 1 of me
1 before you
4 behind you
only see my incoming messages
Belief: Must be 1 + 1 + 4 = 6 of us
ef: t be 1 + 3 = 6 f us

adapted from MacKay (2003) textbook

What about a general DAG?

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree



7 here

3 here

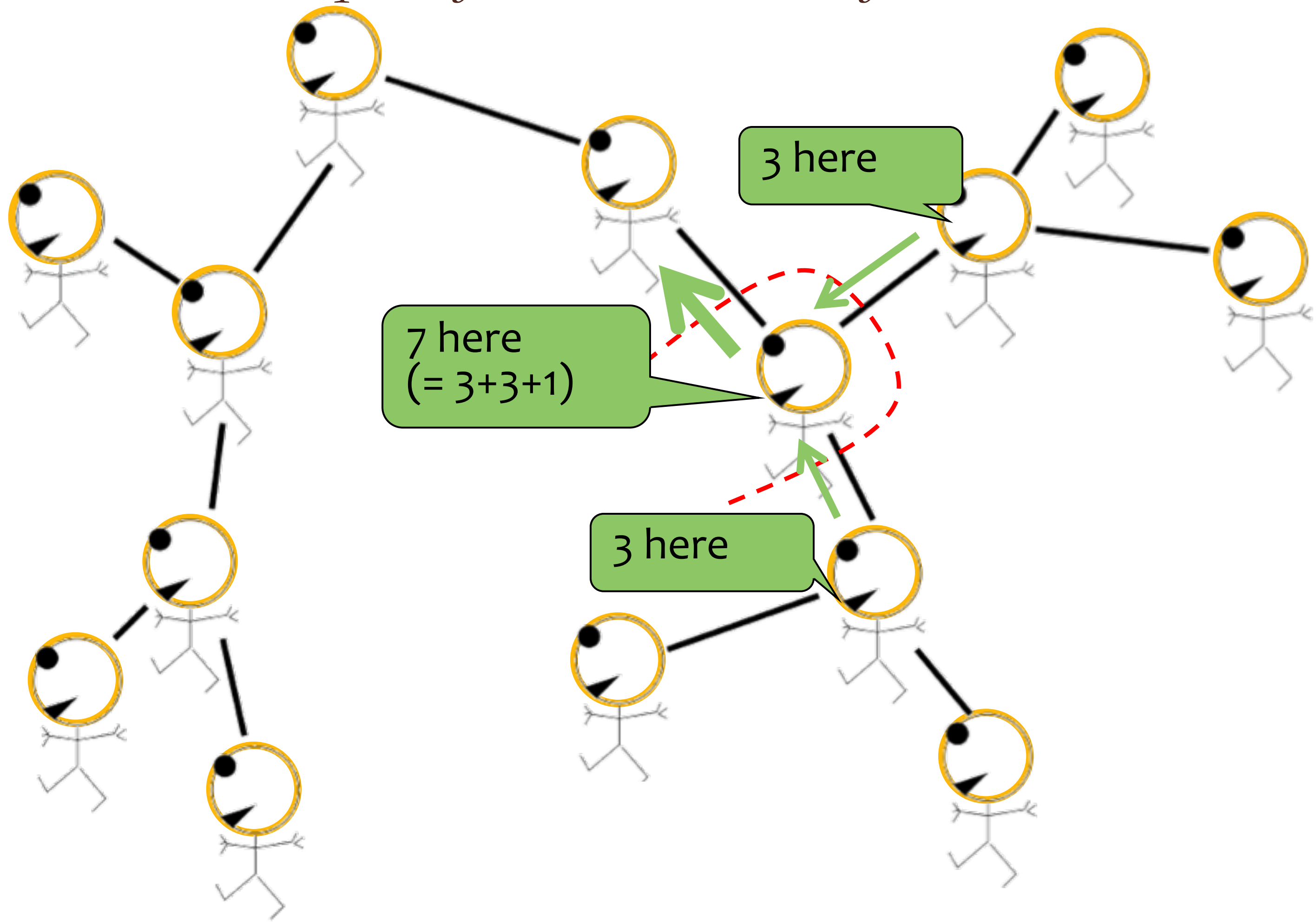
1 of me

11 here
(= 7+3+1)

adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

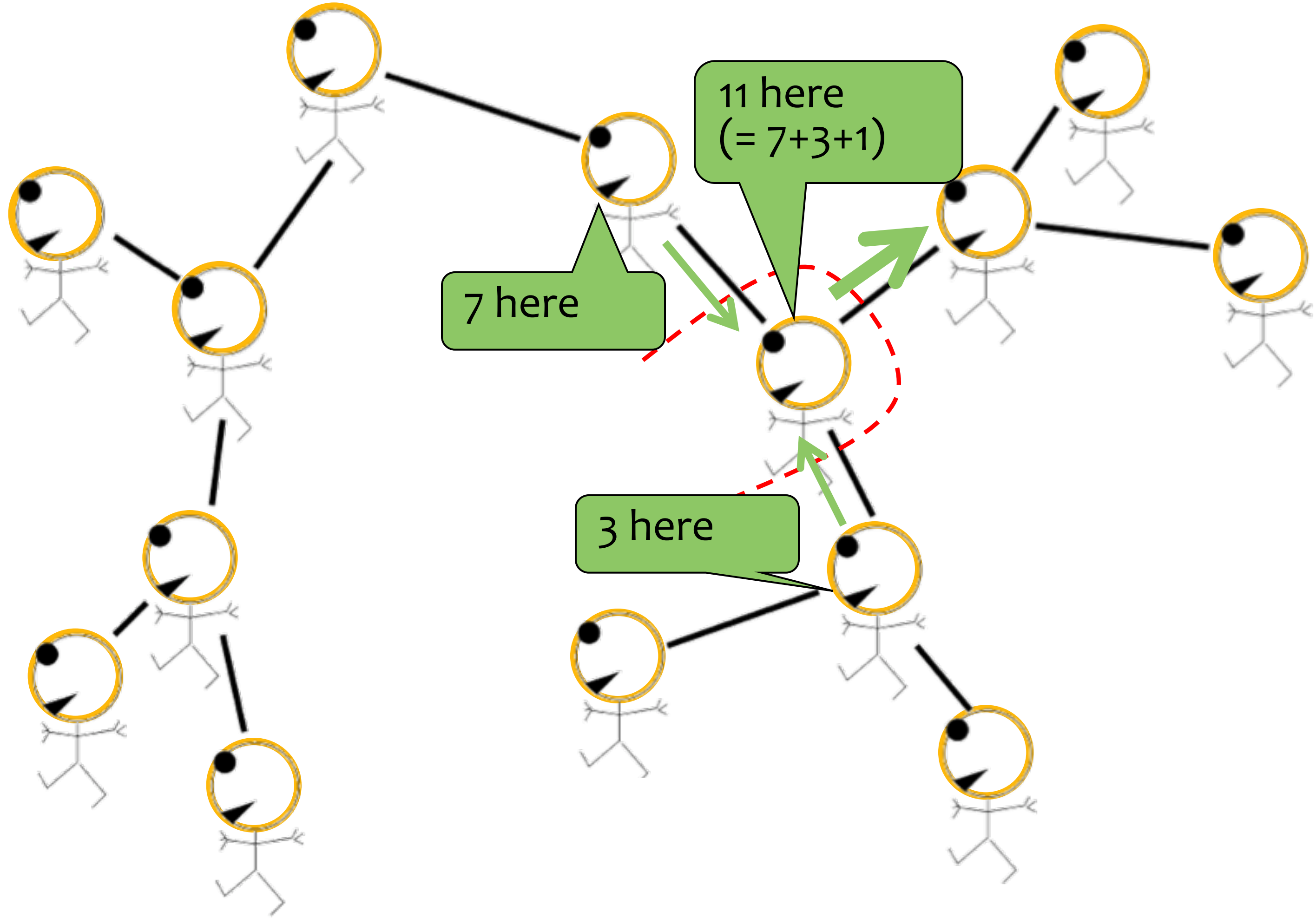
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

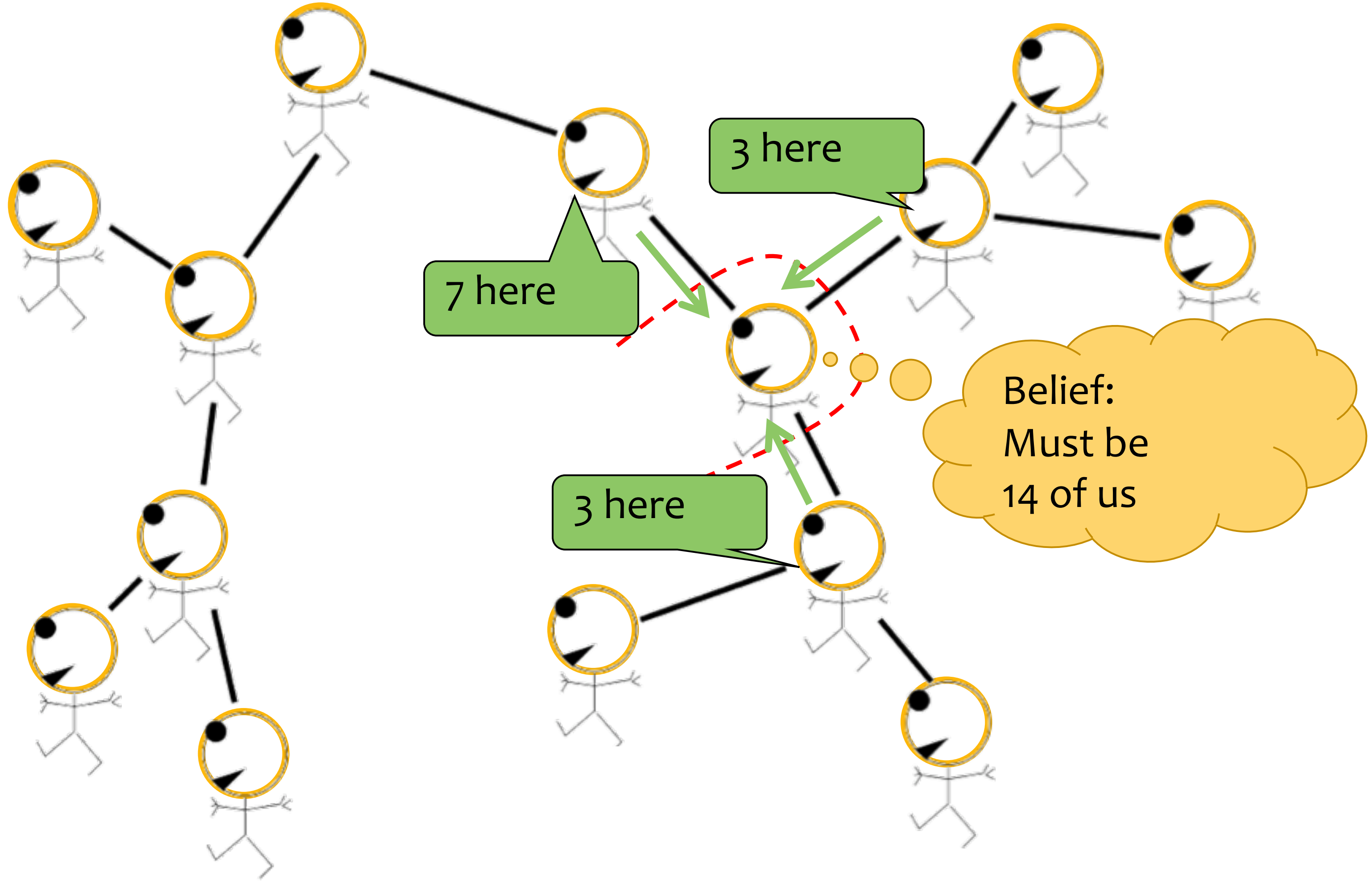
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

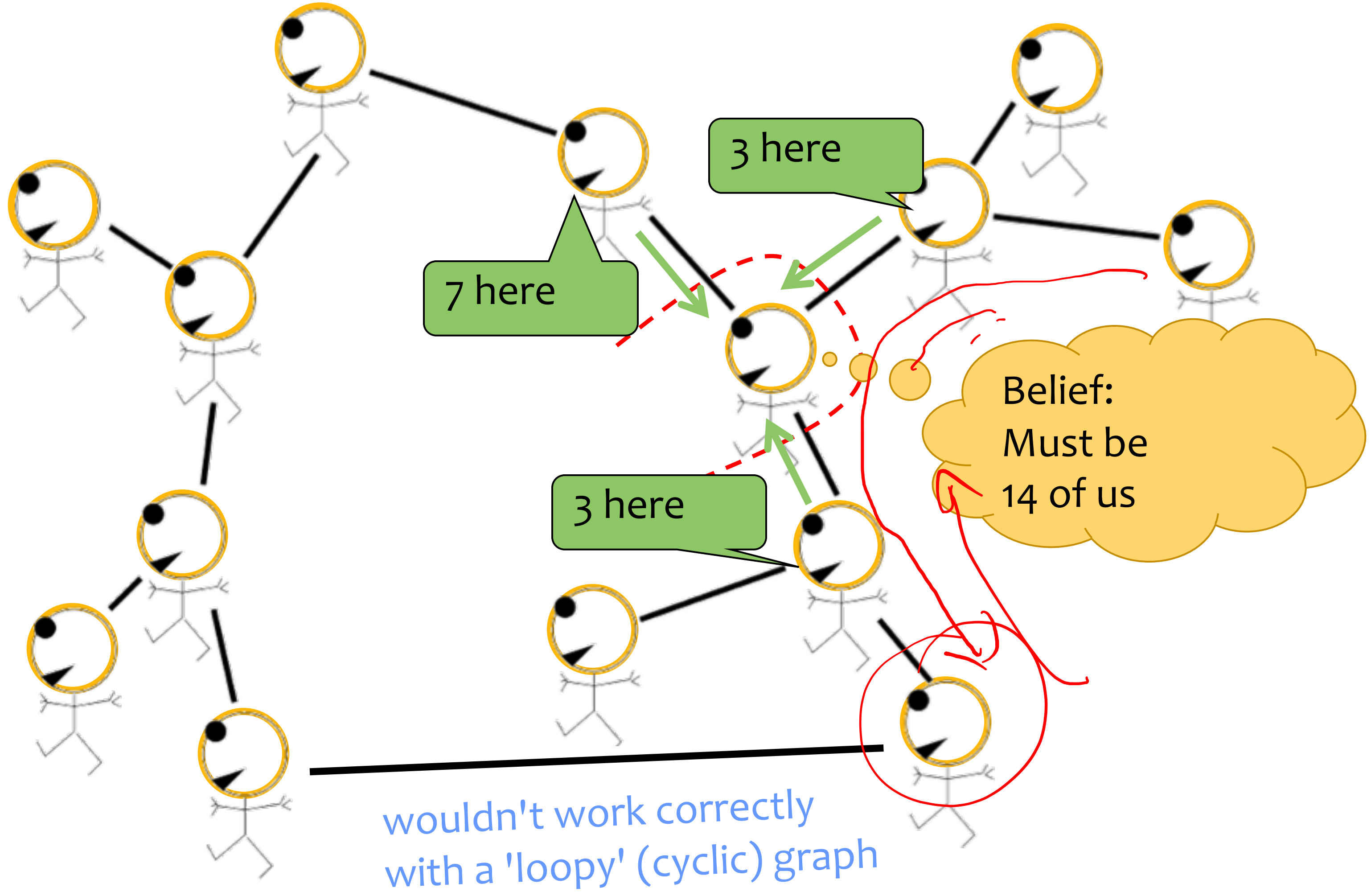
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

Each soldier receives reports from all branches of tree

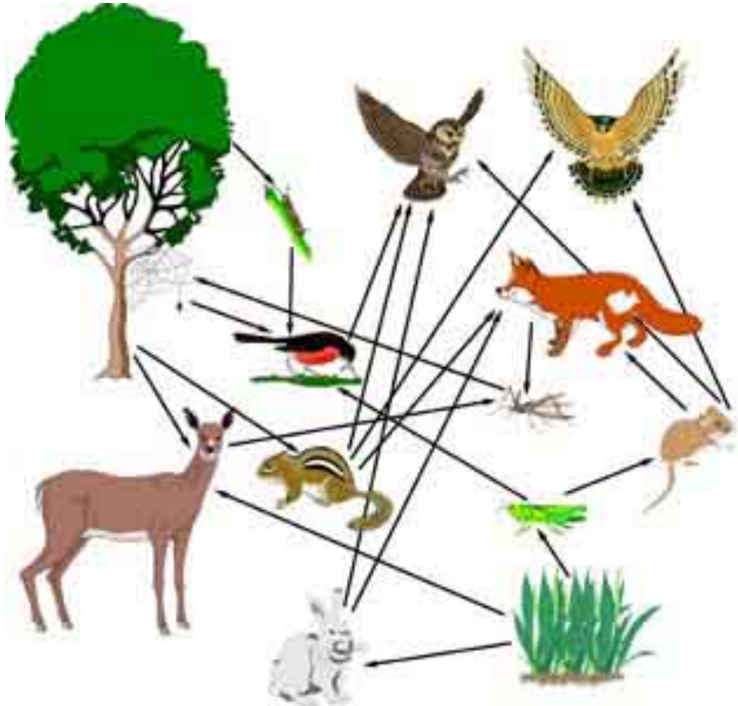


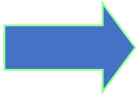
wouldn't work correctly
with a 'loopy' (cyclic) graph

adapted from MacKay (2003) textbook

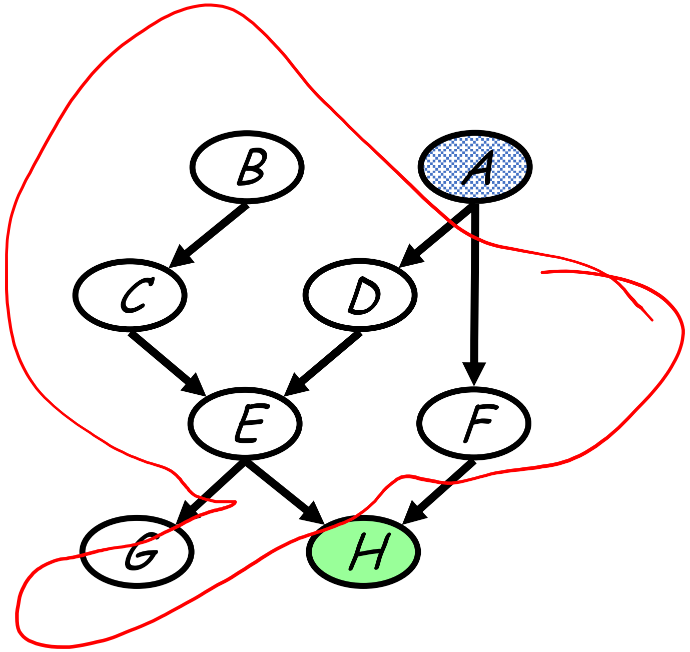
A more complex network

A food web





$P(A \mid H)$



What is the probability that hawks are leaving given that the grass condition is poor?

$m_t(\sim)$

Example: Variable Elimination

- Query: *P(A | h)*
  - Need to eliminate: *B,C,D,E,F,G,H*
- Initial factors:

$$\Sigma\Sigma \sum_{H} P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$

$m_h(e,f)$

$m_h(e,f)$

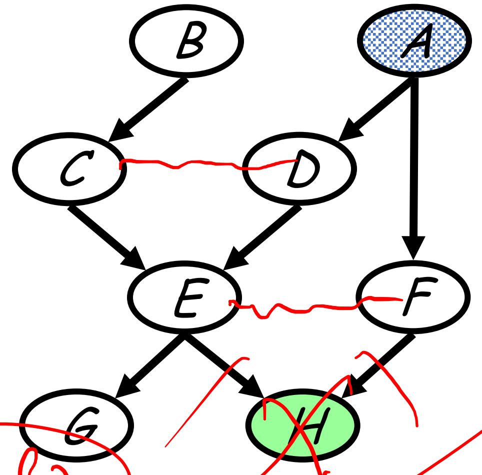
- Choose an elimination order: *H,G,F,E,D,C,B*

- Step 1:
  - **Conditioning** (fix the evidence node (i.e., *h*) on its observed value (i.e., $\tilde{h}$)):

$$m_h(e,f) = p(h = \tilde{h} \mid e,f)$$

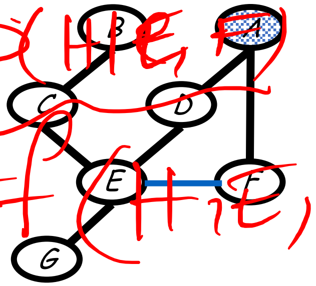
  - This step is isomorphic to a marginalization step:

$$m_h(e,f) = \sum_h p(h \mid e,f)\delta(h = \tilde{h})$$



h

$m(H,E)$



$P(H|E,F)$

$\sum_F f(H,E,F) =$

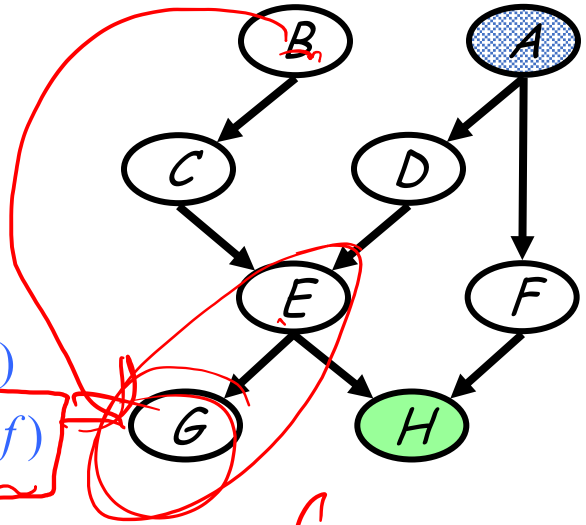
Example: Variable Elimination

- Query: *P(A |h)*
  - Need to eliminate: *B,C,D,E,F,G*

- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)\underset{m_g(e)}{\cancel{P(g|e)}}m_h(e,f)$$

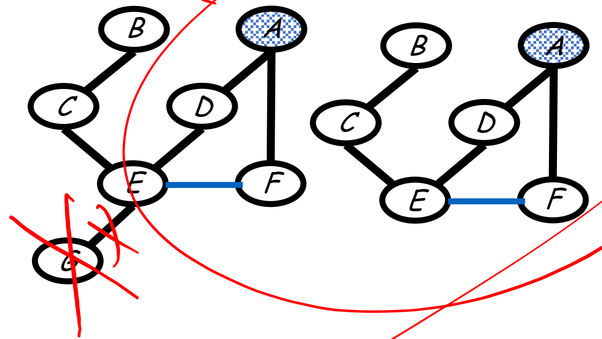


P(g|b~~?~~,e)

$$\sum_{G} f(G,E) = g(E)$$

- Step 2: Eliminate *G*

$$m_g(e) = \underbrace{\sum_g p(g|e)}_{f(g,e)} = 1$$

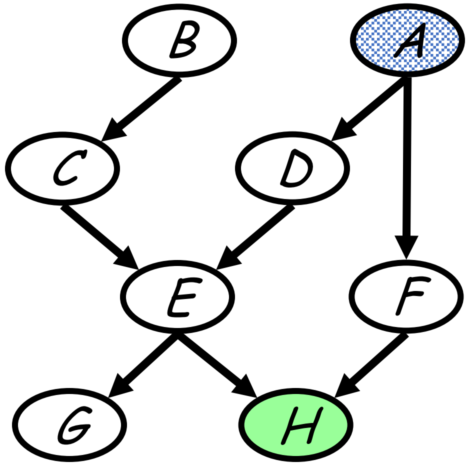


Example: Variable Elimination

- Query: *P(A |h)*
  - Need to eliminate: *B,C,D,E,F*

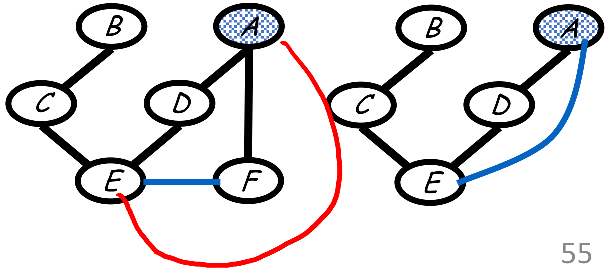
- Initial factors:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)\cancel{P(f|a)m_h(e,f)}$$
$$m_f(e,a)$$



- Step 3: Eliminate *F*

$$m_f(e,a) = \sum_f p(f|a)m_h(e,f)$$



Example: Variable Elimination

- Query: $P(A\,|h)$
  - Need to eliminate: B,C,D,E

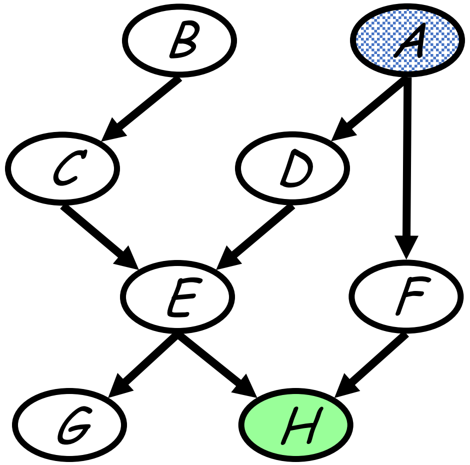
- Initial factors:

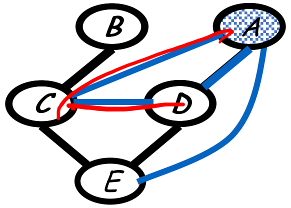
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\cancel{P(e|c,d)m_f(a,e)}$$

$\cancel{m_e(a,c,d)}$ $m(a,c,d)$

- Step 4: Eliminate E

$$m_e(a,c,d)=\sum_e p(e|c,d)m_f(a,e)$$





Example: Variable Elimination

- Query: *P(A | h)*
  - Need to elimінate: *B,C,D*

- Initial factors: $P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$

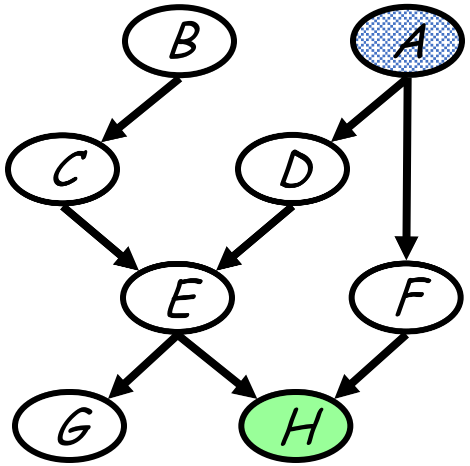
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$

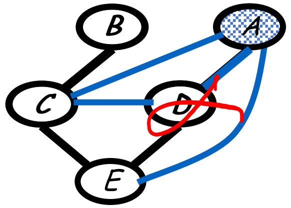
$$\Rightarrow P(a)P(b)P(c|b)\cancel{P(d|a)m_e(a,c,d)}$$

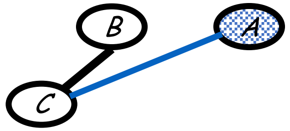
$m_d(a,c)$



- Step 5: Eliminate *D*

$$m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$$





Example: Variable Elimination

- Query: *P(A |h)*
  - Need to eliminate: *B,C*

- Initial factors: $P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$

$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

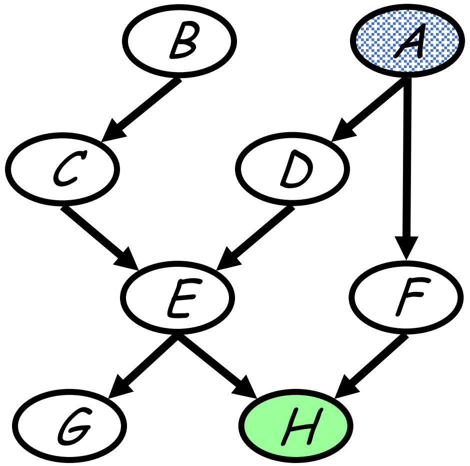
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$

$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$

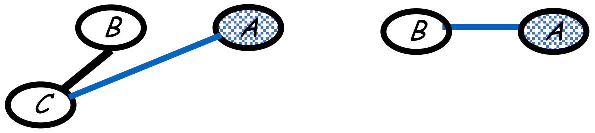
$$\Rightarrow P(a)P(b)\cancel{P(c|d)m_d(a,c)}$$

$$m_c(a,b)$$



- Step 6: Eliminate *C*

$$m_c(a,b) = \sum_c p(c|b)m_d(a,c)$$

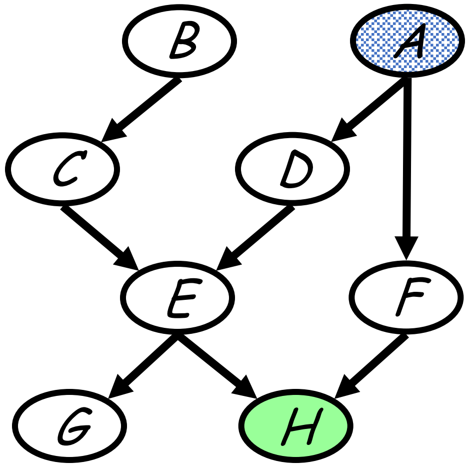


Example: Variable Elimination

- Query: *P(A |h)*
  - Need to eliminate: *B*

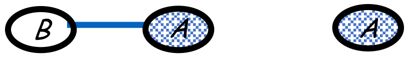
- Initial factors:

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)\underset{m_b(a)}{\cancel{P(b)m_c(a,b)}}$$



- Step 7: Eliminate *B*

$$m_b(a)=\sum_b p(b)m_c(a,b)$$

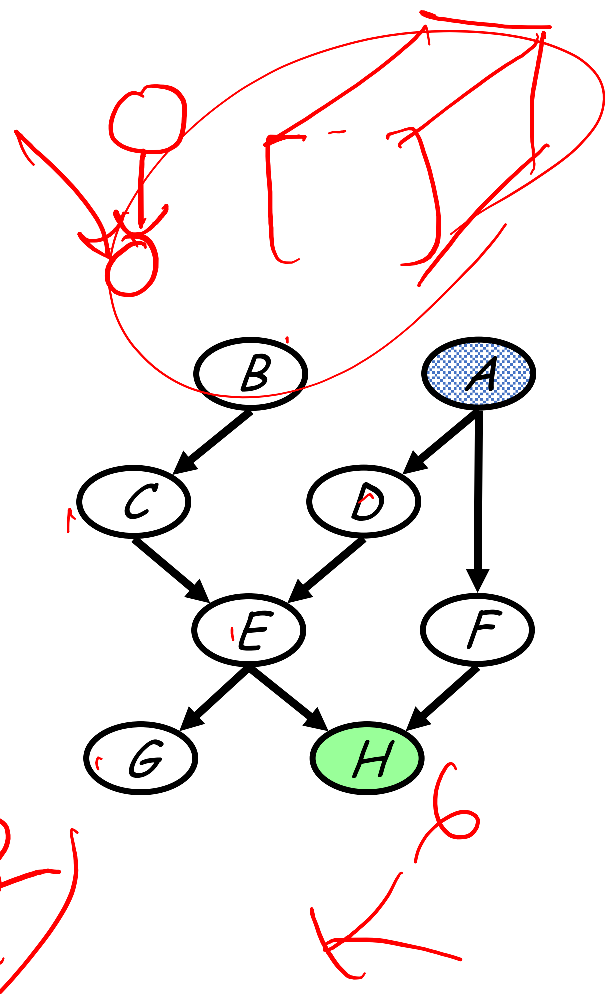


Example: Variable Elimination

- Query: *P(A |h)*
  - Need to eliminate: *B*

- Initial factors:

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$$
$$\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$$
$$\Rightarrow P(a)P(b)m_c(a,b)$$
$$\Rightarrow P(a)m_b(a)$$



$O(6K^B)$

- Step 8: Wrap-up

$$p(a,\tilde{h}) = p(a)m_b(a), \quad p(\tilde{h}) = \sum_a p(a)m_b(a)$$

$$\Rightarrow P(a|\tilde{h}) = \frac{p(a)m_b(a)}{\sum_a p(a)m_b(a)}$$

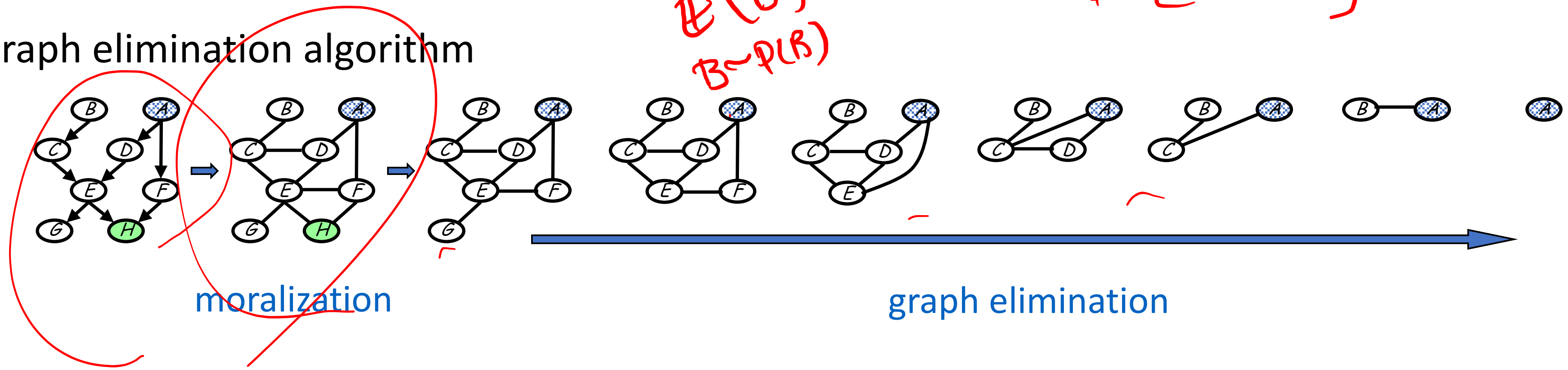
~~$O(K^2)$~~

© Eric Xing @ CMU, 2005-2015

Understanding Variable Elimination

$\mathbb{E}_{B \sim p(B)}[B] = \qquad \mathbb{E}[\text{—}]$

- A graph elimination algorithm

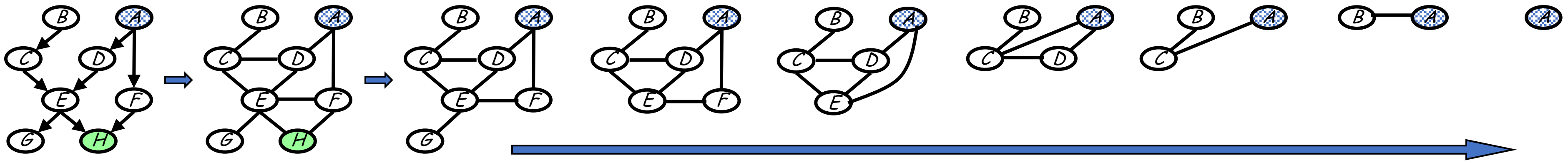


Graph elimination

- Begin with the undirected GM or moralized BN
- Graph $G(V, E)$ and elimination ordering I
- Eliminate next node in the ordering I
  - Removing the node from the graph
  - Connecting the remaining neighbors of the nodes
- The reconstituted graph $G'(V, E')$
  - Retain the edges that were created during the elimination procedure
  - The graph-theoretic property: the factors resulted during variable elimination are captured by recording the elimination clique

Understanding Variable Elimination

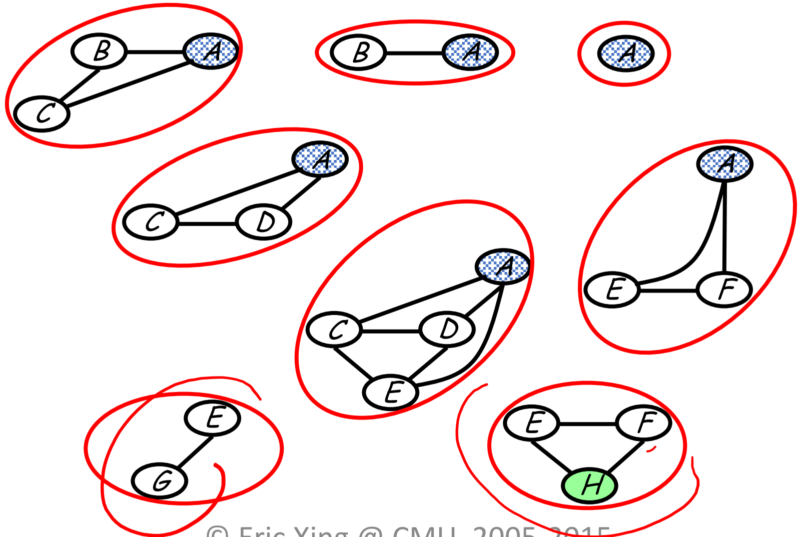
- A graph elimination algorithm



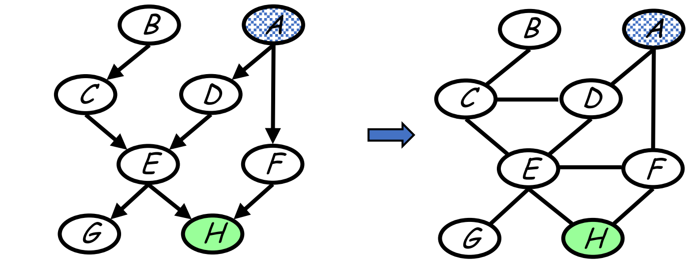
moralization

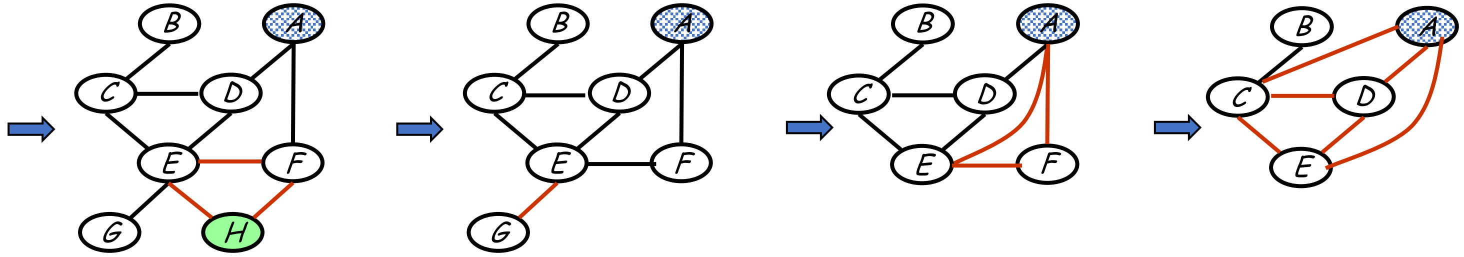
graph elimination

- Intermediate terms correspond to the cliques resulted from elimination

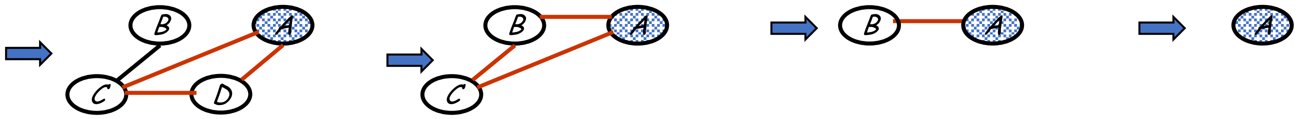


Elimination Cliques





Messages sent: $m_h(e,f)$ $m_g(e)$ $m_f(e,a)$ $m_e(a,c,d)$



Messages sent: $m_d(a,c)$ $m_c(a,b)$ $m_b(a)$

Graph elimination and marginalization

- Induced dependency during marginalization vs. elimination clique
  - Summation <-> elimination
  - Intermediate term <-> elimination clique

$$P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$

1 $\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$

2 $\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$

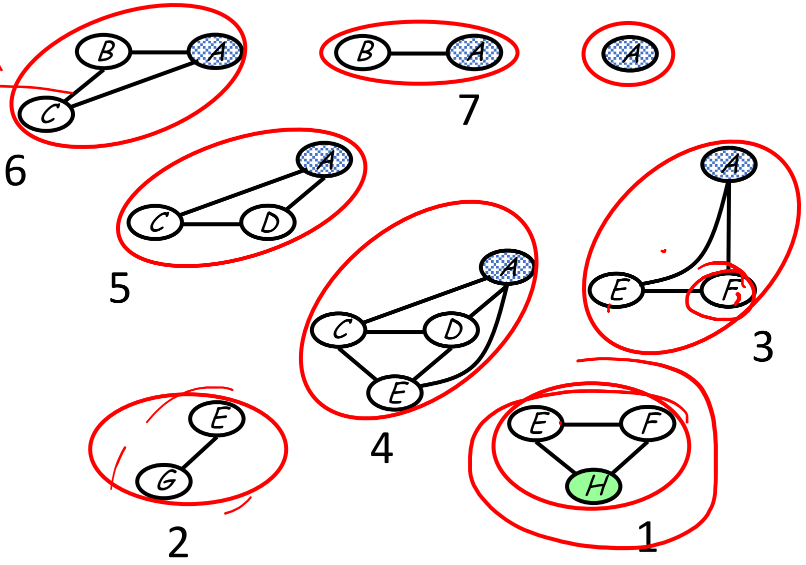
3 $\Rightarrow P(a)P(b)P(c|d)P(d|a)P(e|c,d)m_f(a,e)$

4 $\Rightarrow P(a)P(b)P(c|d)P(d|a)m_e(a,c,d)$

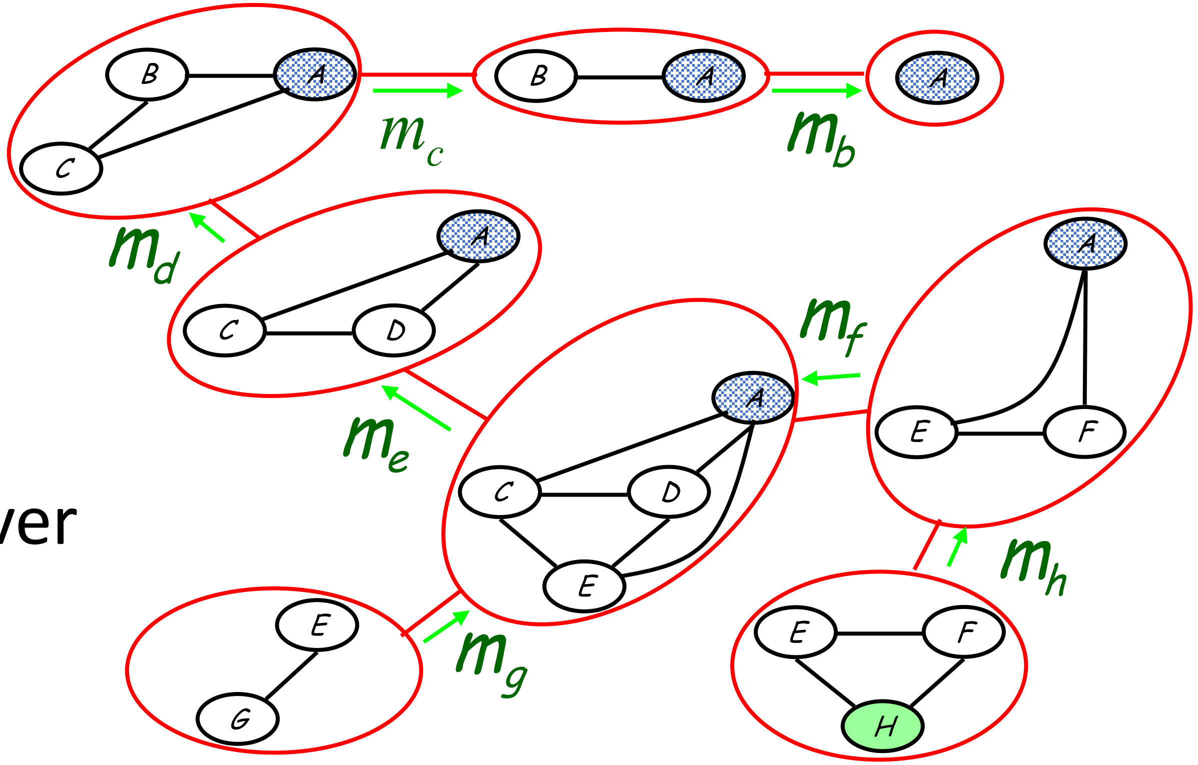
5 $\Rightarrow P(a)P(b)P(c|d)m_d(a,c)$

6 $\Rightarrow P(a)P(b)m_c(a,b)$

7 $\Rightarrow P(a)m_b(a)$



A clique tree



Message from one C1 to C2:

Multiply all incoming messages with the local factor and **sum** over variables that are not shared

$$m_e(a,c,d) = \sum_e p(e|c,d)m_g(e)m_f(a,e)$$

Complexity

k^2 k^3 K^{nr}

- The overall complexity is determined by the number of the largest elimination clique
  - What is the largest elimination clique? – a pure graph theoretic question

  - **Tree-width** *k*: one less than the smallest achievable value of the cardinality of the largest elimination clique, ranging over all possible elimination ordering

  - “good” elimination orderings lead to **small cliques** and hence reduce complexity (what will happen if we eliminate "e" first in the above graph?)

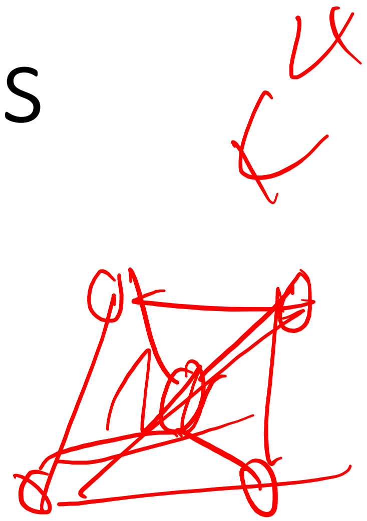
  - Find the best elimination ordering of a graph --- NP-hard

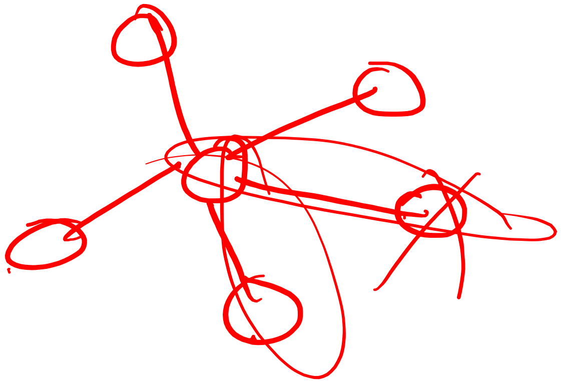
  → Inference is NP-hard

  - But there often exist "obvious" optimal or near-opt elimination ordering

Examples

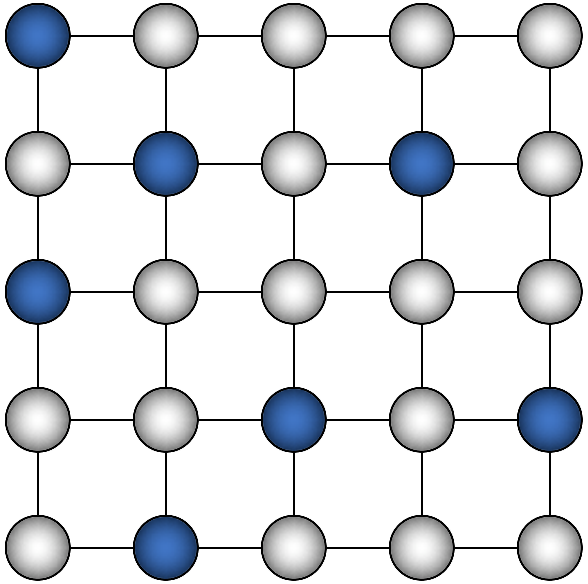
- Star





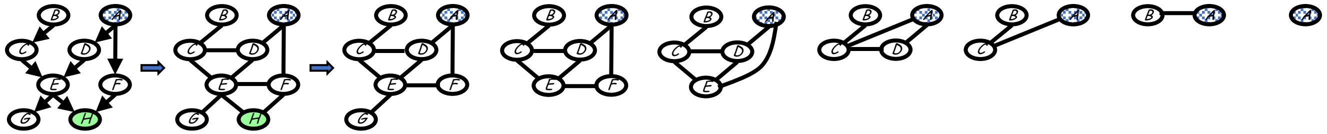
- Tree

More example: Ising model



Limitation of Procedure Elimination

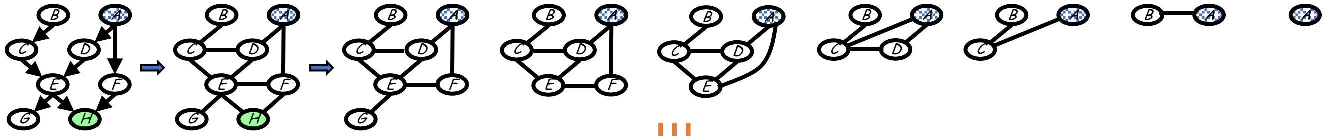
- Limitation

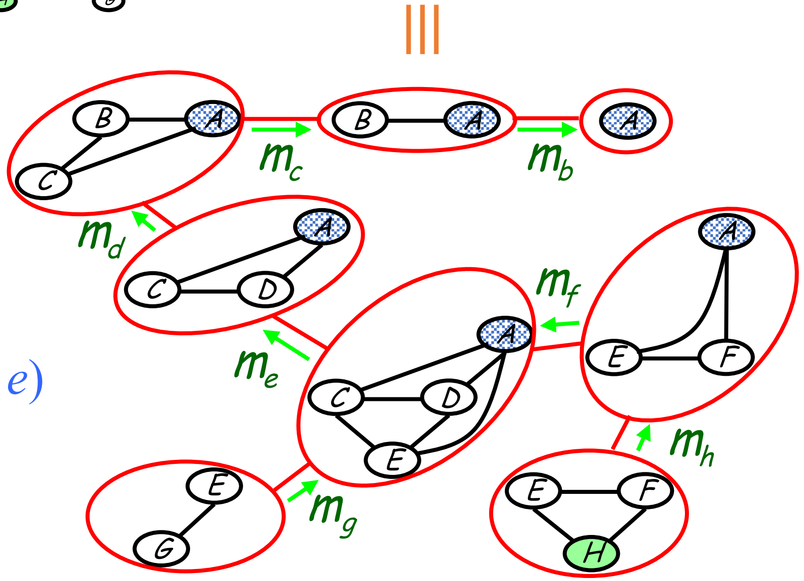


From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?

- Elimination ≡ message passing on a **clique tree**



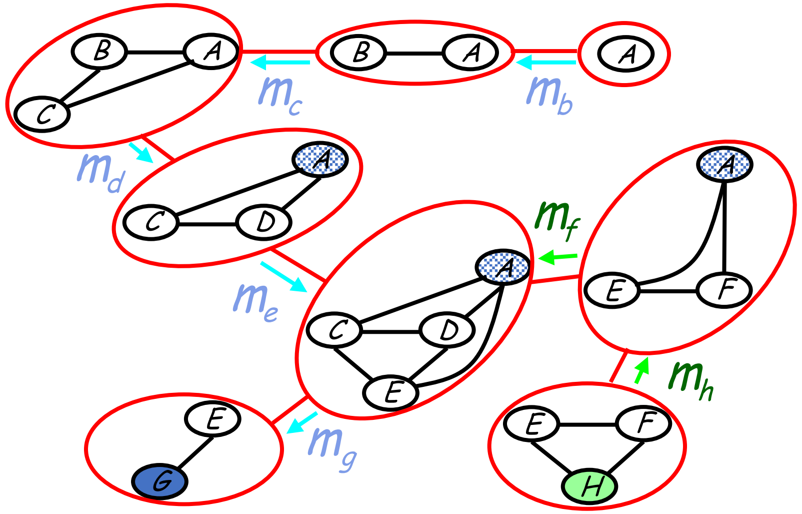


$$m_e(a,c,d) \\ = \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

- Messages can be reused

From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination ≡ message passing on a **clique tree**
  - **Another query ...**



- Messages m_f and m_h are reused, others need to be recomputed

Summary

- The simple Eliminate algorithm captures the key algorithmic Operation underlying probabilistic inference:

  --- That of taking a sum over product of potential functions

- What can we say about the overall computational complexity of the algorithm? In particular, how can we control the "size" of the summands that appear in the sequence of summation operation.

- The computational complexity of the Eliminate algorithm can be reduced to purely graph-theoretic considerations.

- This graph interpretation will also provide hints about how to design improved inference algorithm that overcome the limitation of Eliminate.