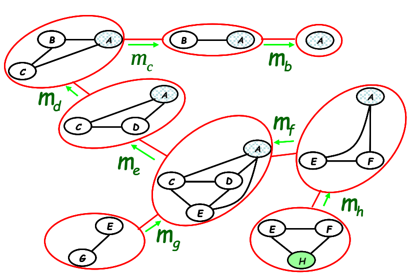
School of Computer Science
Carnegie Mellon

Probabilistic Graphical Models

Exact Inference:

Variable Elimination



Eric Xing

Lecture 4, January 27, 2014

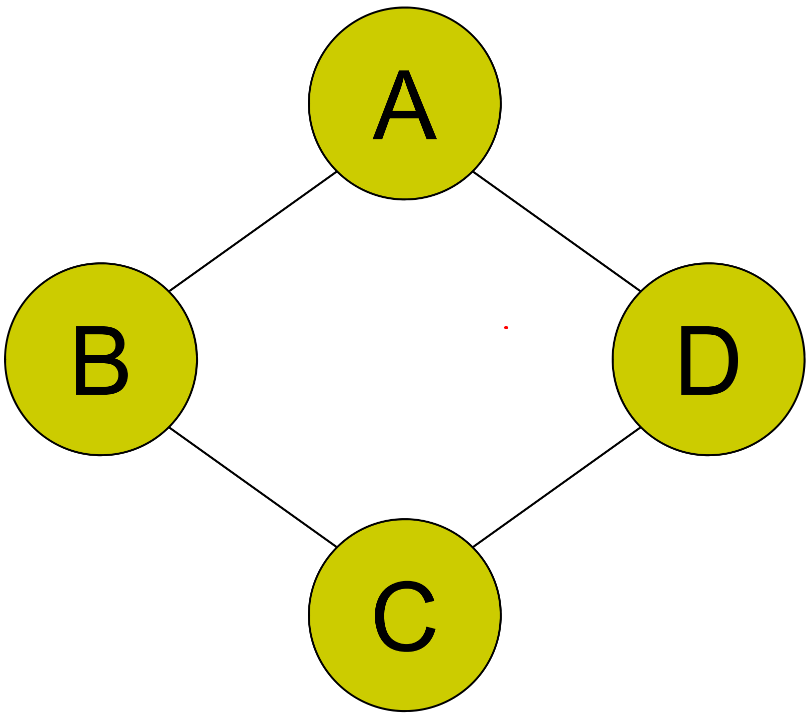
Reading: KF-chap 9

Recap:

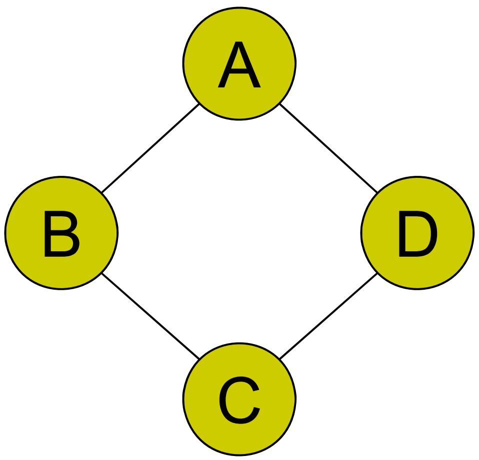
- Defn: A DAG G is a **perfect map** (P-map) for a distribution P if $I(P)=I(G)$.

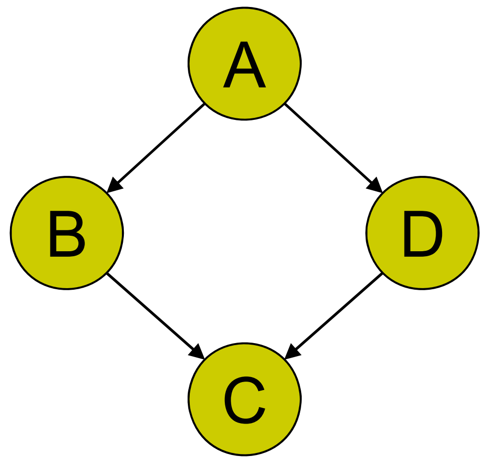
Question: Is there a BN that is a perfect map for a given MN?

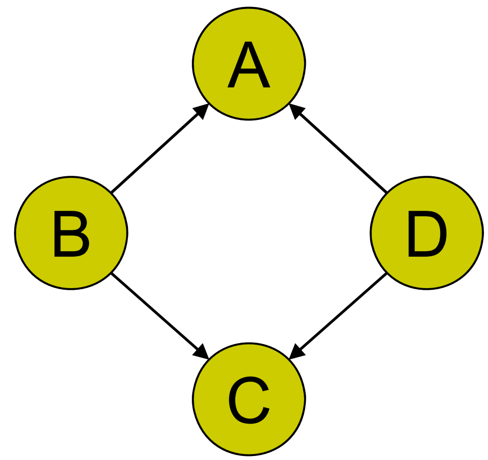
- The "diamond" MN



Question: Is there a BN that is a perfect map for a given MN?







$A \perp C \mid \{B,D\}$

$B \perp D \mid \{A,C\}$

$A \perp C \mid \{B,D\}$

$B \perp D \mid A$

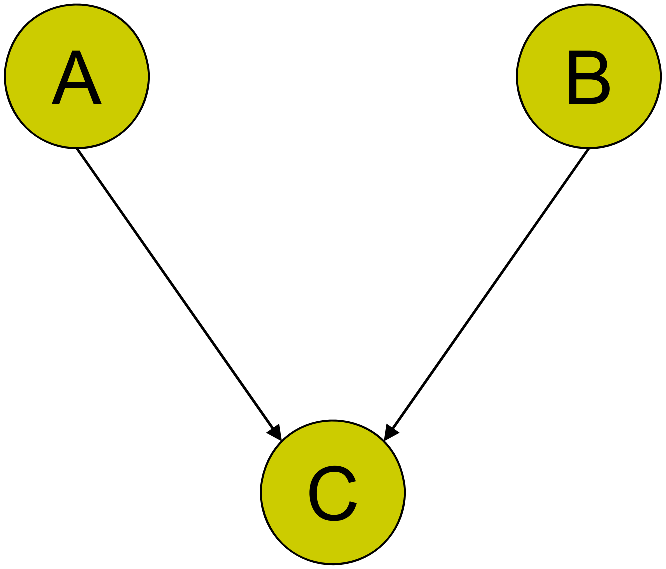
$A \perp C \mid \{B,D\}$

$B \perp D$

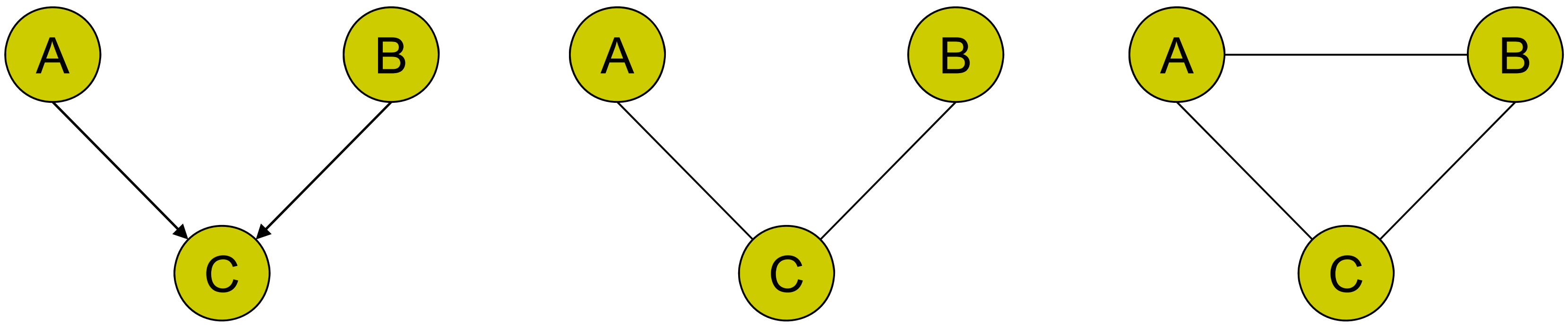
- This MN does not have a perfect I-map as BN!

Question: Is there an MN that is a perfect I-map to a given BN?

- V-structure example



Question: Is there an MN that is a perfect I-map to a given BN?



$A \perp B$

$\neg (A \perp B \mid C)$

$A \perp B \mid C$

$\neg (A \perp B)$

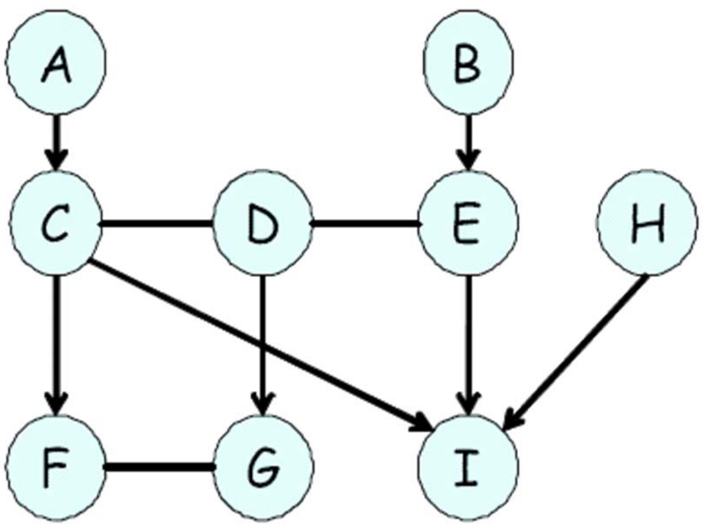
$\neg (A \perp B \mid C)$

$\neg (A \perp B)$

- V-structure has no equivalent in MNs!

Partially Directed Acyclic Graphs

- Also called chain graphs
- Nodes can be disjointly partitioned into several chain components
- An edge within the same chain component must be undirected
- An edge between two nodes in different chain components must be directed



Chain components:

{A}, {B}, {C,D,E},{F,G},{H}, {I}

Summary

- Investigated the relationship between BNs and MNs
  - They represent different families of independence assumptions

- Not mentioned: Chain networks → superset of both BNs and MNs

- Why we care about this:
  - BN and MN offer different semantics for designer to capture or expression (conditional) independences among variables
  - Under certain condition BN can be represented as an MN and vice versa
  - In the future, for certain operation (i.e., inference), we will be using a single representation as the "data structure" for which an algorithm can operate on.
  - This makes algorithm design, and analysis of the algorithms simpler

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*?
    i. We use **learning** as a name for the process of obtaining point estimate of *M*.
    ii. But for *Bayesian*, they seek $p(M|D)$, which is actually an **inference** problem.
    iii. When not all variables are observable, even computing point estimate of *M* need to do **inference** to impute the *missing data*.

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

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

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

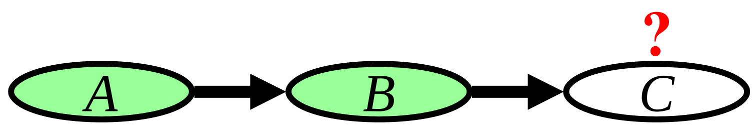
- We usually query a subset $\mathbf{Y}$ of all domain variables $\mathbf{X}=\{\mathbf{Y},\mathbf{Z}\}$ and "don't care" about the remaining, $\mathbf{Z}$:

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

  - the process of summing out the "don't care" variables z is called **marginalization**, and the resulting $P(y|e)$ is called a **marginal** prob.

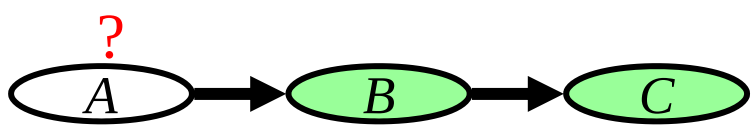
Applications of a posteriori Belief

- **Prediction**: what is the probability of an outcome given the starting condition



  - the query node is a descendent of the evidence

- **Diagnosis**: what is the probability of disease/fault given symptoms



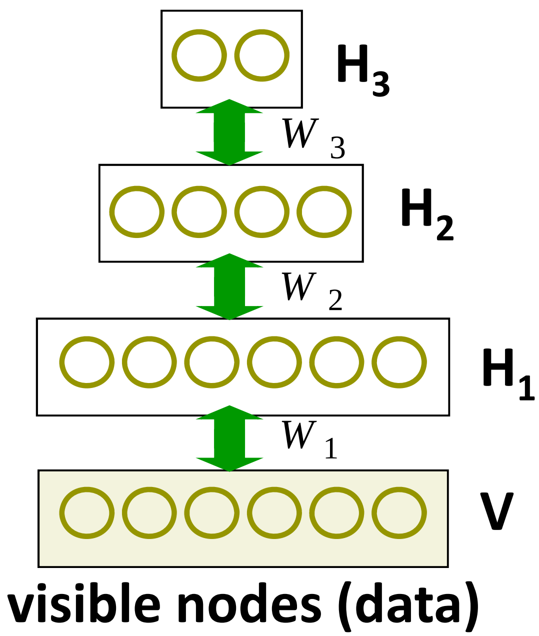
  - the query node an ancestor of the evidence

- **Learning** under partial observation
  - fill in the unobserved values under an "EM" setting (more later)

- The directionality of information flow between variables is not restricted by the directionality of the edges in a GM
  - probabilistic inference can combine evidence form all parts of the network

Example: Deep Belief Network

- Deep Belief Network (DBN) [Hinton et al., 2006]
  - Generative model with multiple hidden layers
  - Successful applications
    - Recognizing handwritten digits
    - Learning motion capture data
    - Collaborative filtering



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 $\mathbf{e}$, and ignoring (the values of) other variables $\mathbf{z}$:

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

  - this is the **maximum *a posteriori*** configuration of $\mathbf{y}$.

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) ?

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

Complexity of Inference

Thm:

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

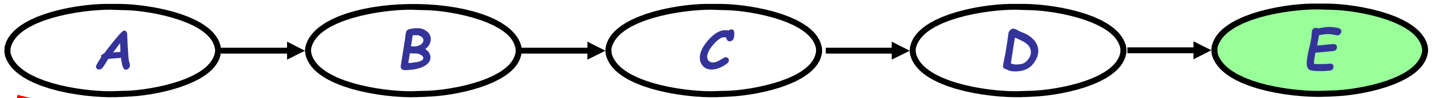
- Hardness does not mean we cannot solve inference
  - It implies that we cannot find a general procedure that works efficiently for arbitrary GMs
  - For particular families of GMs, we can have provably efficient procedures

Approaches to inference

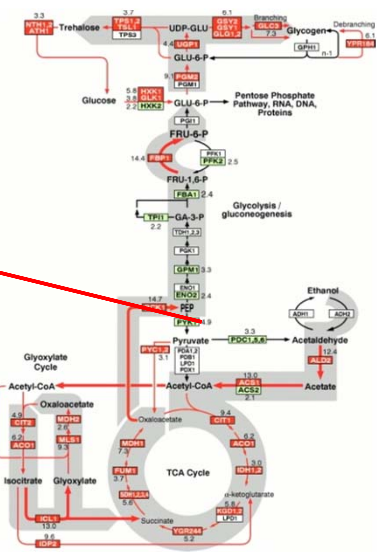
- Exact inference algorithms
  - The elimination algorithm
  - Message-passing algorithm (sum-product, belief propagation)
  - The junction tree algorithms
- Approximate inference techniques
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
  - Variational algorithms

Marginalization and Elimination

- A signal transduction pathway:



What is the likelihood that protein E is active?



- Query: $P(e)$

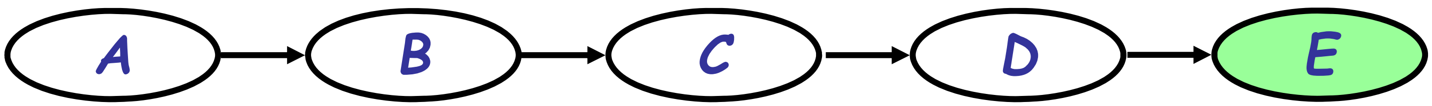
$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a,b,c,d,e)$$

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

- By chain decomposition, we get

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

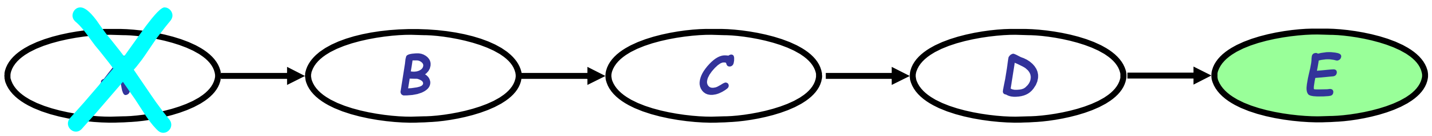
Elimination on Chains



- Rearranging terms ...

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

Elimination on Chains

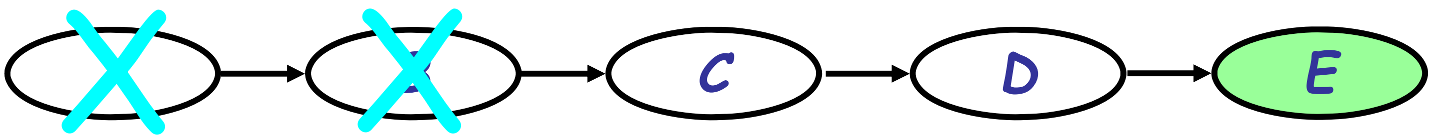


- Now we can perform innermost summation

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

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

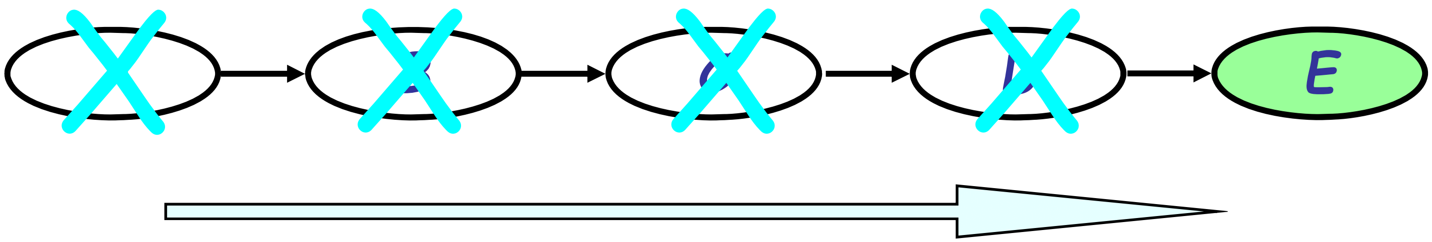
Elimination in Chains



- Rearranging and then summing again, we get

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

Elimination in Chains

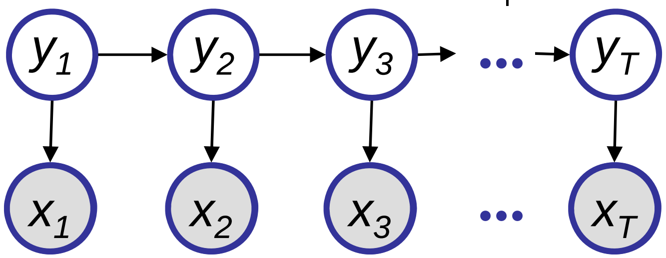


- Eliminate nodes one by one all the way to the end, we get

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

- Complexity:
  - Each step costs $O(|Val(X_i)|*|Val(X_{i+1})|)$ operations: $O(kn^2)$
  - Compare to naïve evaluation that sums over joint values of n-1 variables $O(n^k)$

Hidden Markov Model

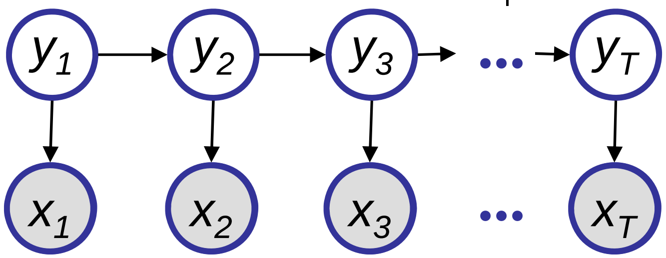


$$
\begin{aligned}
p(\mathbf{x}, \mathbf{y}) &= p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \\
&= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)
\end{aligned}
$$

Conditional probability:

$$
\begin{aligned}
p(y_i | x_1, \ldots, x_T) &= \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} p(y_i, \ldots, y_T, x_1, \ldots, x_T) \\
&= \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} p(y_1) p(x_1 | y_1) \ldots p(y_T | y_{T-1}) p(x_T | y_T)
\end{aligned}
$$

Hidden Markov Model

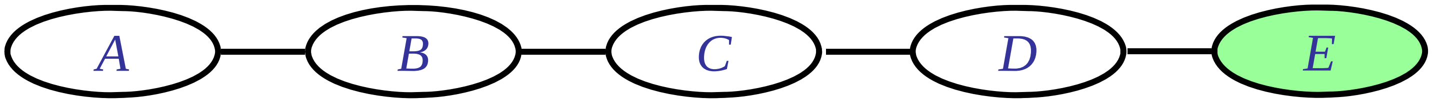


Conditional probability:

$$p(y_i|x_1,\ldots,x_T) = \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} p(y_i,\ldots,y_T,x_1,\ldots,x_T)$$

$$= \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} p(y_1)p(x_1|y_1)\ldots p(y_T|y_{T-1})p(x_T|y_T)$$

Undirected Chains



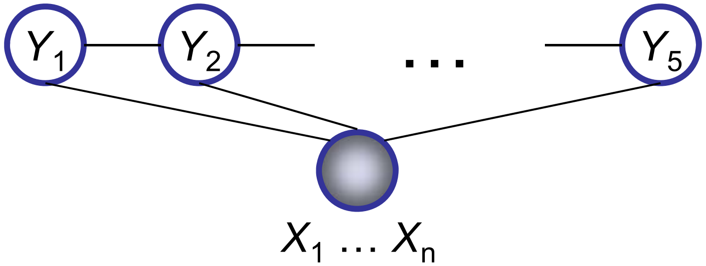
- Rearranging terms ...

$$P(e) = \sum_d \sum_c \sum_b \sum_a \frac{1}{Z} \phi(b,a)\phi(c,b)\phi(d,c)\phi(e,d)$$

$$= \frac{1}{Z} \sum_d \sum_c \sum_b \phi(c,b)\phi(d,c)\phi(e,d) \sum_a \phi(b,a)$$

$$= \cdots$$

Conditional Random Fields



$$
\begin{aligned}
& p(y_1, \ldots, y_T | \mathbf{x}) \\
\propto \quad & \exp\Big(w_{1,2}\phi(y_1, y_2|\mathbf{x}) + w_{2,3}\phi(y_2, y_3|\mathbf{x}) + \ldots + w_{T-1,T}\phi(y_{T-1}, y_T|\mathbf{x}) \\
& + u_1\phi(y_1|\mathbf{x}) + u_2\phi(y_2|\mathbf{x}) + \ldots + u_T\phi(y_T|\mathbf{x}) \Big) \\
= \quad & \Phi(y_1, y_2)\Phi(y_2, y_3) \ldots \Phi(y_{T-1}, y_T)
\end{aligned}
$$

The Sum-Product Operation

- In general, we can view the task at hand as that of computing the value of an expression of the form:

$$\sum_{\mathbf{z}} \prod_{\phi \in \mathcal{F}} \phi$$

where $\mathcal{F}$ is a set of factors

- We call this task the *sum-product* inference task.

Inference on General GM via Variable Elimination

General idea:

- Write query in the form

$$P(X_1, \mathbf{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

  - this suggests an "elimination order" of latent variables to be marginalized

- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product

- wrap-up

$$P(X_1 \mid \mathbf{e}) = \frac{\phi(X_1, \mathbf{e})}{\sum_{x_1} \phi(X_1, \mathbf{e})}$$

Outcome of elimination

- Let $\mathbf{X}$ be some set of variables,
  let $\mathcal{F}$ be a set of factors such that for each $\phi \in \mathcal{F}$, *Scope*$[\phi] \in \mathbf{X}$,
  let $\mathbf{Y} \subset \mathbf{X}$ be a set of query variables,
  and let $Z = \mathbf{X} - \mathbf{Y}$ be the variable to be eliminated

- The result of eliminating the variable Z is a factor

$$\tau(\mathbf{Y}) = \sum_{\mathbf{z}} \prod_{\phi \in \mathcal{F}} \phi$$

  - This factor does not necessarily correspond to any probability or conditional probability in this network. (example forthcoming)

Dealing with evidence

- Conditioning as a Sum-Product Operation
  - The evidence potential:
    $$\delta(E_i, \overline{e}_i) = \begin{cases} 1 & \text{if } E_i \equiv \overline{e}_i \\ 0 & \text{if } E_i \neq \overline{e}_i \end{cases}$$
  - Total evidence potential:
    $$\delta(\mathbf{E}, \overline{\mathbf{e}}) = \prod_{i \in I_{\mathbf{E}}} \delta(E_i, \overline{e}_i)$$
  - Introducing evidence --- restricted factors:

$$\tau(\mathbf{Y}, \overline{\mathbf{e}}) = \sum_{\mathbf{z},\mathbf{e}} \prod_{\phi \in \mathcal{F}} \phi \times \delta(\mathbf{E}, \overline{\mathbf{e}})$$

The elimination algorithm

Procedure Elimination (

G, // the GM

$\mathbf{E}$, // evidence

Z, // Set of variables to be eliminated

X, // query variable(s)

)

1. Initialize (G)
2. Evidence ($\mathbf{E}$)
3. Sum-Product-Elimination ($\mathscr{F}$, Z, $\prec$)
4. Normalization ($\mathscr{F}$)

The elimination algorithm

Procedure Initialize (G, Z)

1. Let $Z_1, \ldots, Z_k$ be an ordering of Z such that $Z_i \prec Z_j$ iff $i < j$
2. Initialize $\mathcal{F}$ with the full the set of factors

Procedure Evidence $(\mathbf{E})$

1. **for** each $i \in I_E$,
   $\mathcal{F} = \mathcal{F} \cup \delta(E_i, e_i)$

Procedure Sum-Product-Variable-Elimination $(\mathcal{F}, Z, \prec)$

1. **for** $i = 1, \ldots, k$
   $\mathcal{F} \leftarrow$ Sum-Product-Eliminate-Var$(\mathcal{F}, Z_i)$
2. $\phi^* \leftarrow \prod_{\phi \in \mathcal{F}} \phi$
3. **return** ϕ^*
4. Normalization (ϕ^*)

The elimination algorithm

Procedure Initialize (G, Z)

1. Let $Z_1, \ldots, Z_k$ be an ordering of Z such that $Z_i \prec Z_j$ iff $i < j$
2. Initialize $\mathcal{F}$ with the full the set of factors

Procedure Evidence $(\mathbf{E})$

1. **for** each $i \in I_E$,
   $\mathcal{F} = \mathcal{F} \cup \delta(E_i, e_i)$

Procedure Sum-Product-Variable-Elimination $(\mathcal{F}, Z, \prec)$

1. **for** $i = 1, \ldots, k$
   $\mathcal{F} \leftarrow$ Sum-Product-Eliminate-Var$(\mathcal{F}, Z_i)$
2. $\phi^* \leftarrow \prod_{\phi \in \mathcal{F}} \phi$
3. **return** ϕ^*
4. Normalization (ϕ^*)

Procedure Normalization (ϕ^*)

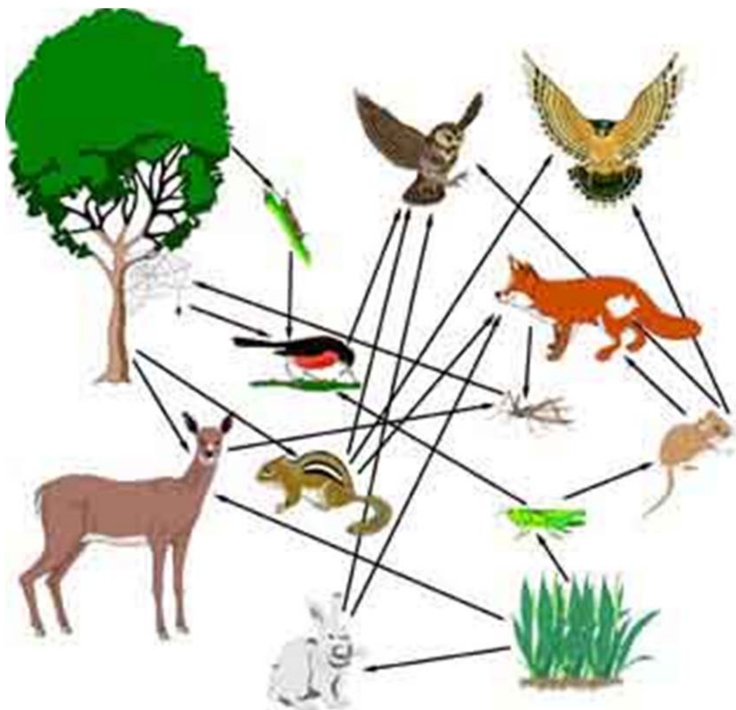
1. $P(X|\mathbf{E}) = \phi^*(X) / \sum_x \phi^*(X)$

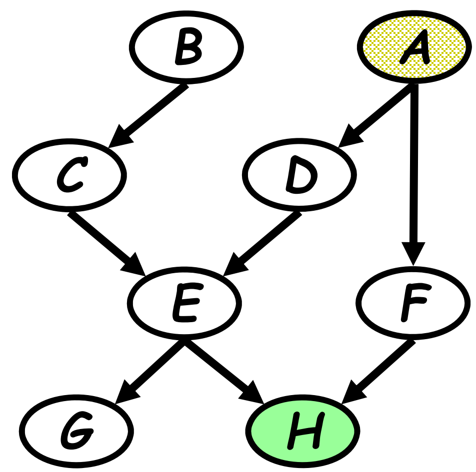
Procedure Sum-Product-Eliminate-Var (
    $\mathcal{F}$, // Set of factors
    Z // Variable to be eliminated
)

1. $\mathcal{F}' \leftarrow \{\phi \in \mathcal{F} : Z \in Scope[\phi]\}$
2. $\mathcal{F}'' \leftarrow \mathcal{F} - \mathcal{F}'$
3. $\psi \leftarrow \prod_{\phi \in \mathcal{F}'} \phi$
4. $\tau \leftarrow \sum_Z \psi$
5. **return** $\mathcal{F}'' \cup \{\tau\}$

A more complex network

A food web

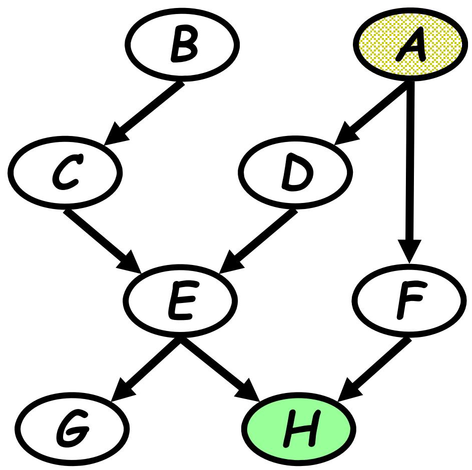




What is the probability that hawks are leaving given that the grass condition is poor?

Example: Variable Elimination

- Query: $P(A \mid h)$
  - Need to eliminate: *B,C,D,E,F,G,H*



- Initial factors:

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

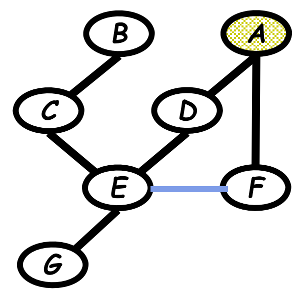
- Choose an elimination order: *H,G,F,E,D,C,B*

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

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

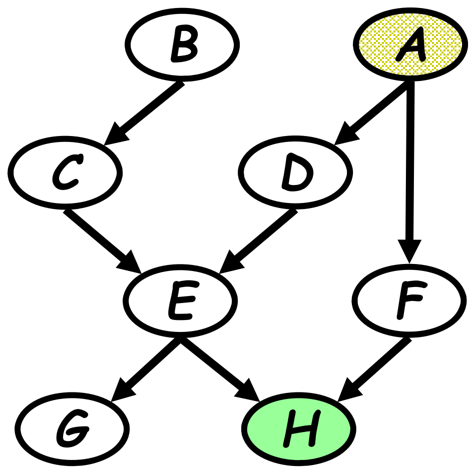
  - This step is isomorphic to a marginalization step:

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



Example: Variable Elimination

- Query: $P(B \mid h)$
  - Need to eliminate: B,C,D,E,F,G

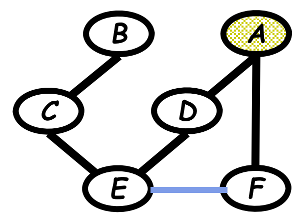


- Initial factors:

$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)\underline{P(g \mid e)}m_h(e,f)$$

- Step 2: Eliminate G
  - compute

$$m_g(e) = \sum_g p(g \mid e) = 1$$

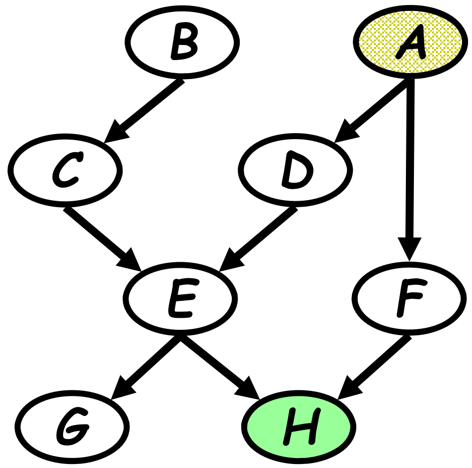


$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_g(e)m_h(e,f)$$
$$= P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)\underline{m_h(e,f)}$$

Example: Variable Elimination

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

- Initial factors:

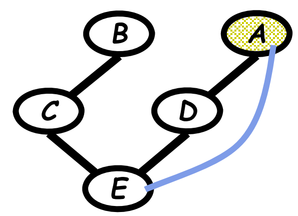


$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)\underline{P(f \mid a)m_h(e,f)}$$

- Step 3: Eliminate F
  - compute

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

$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)\underline{m_f(a,e)}$$

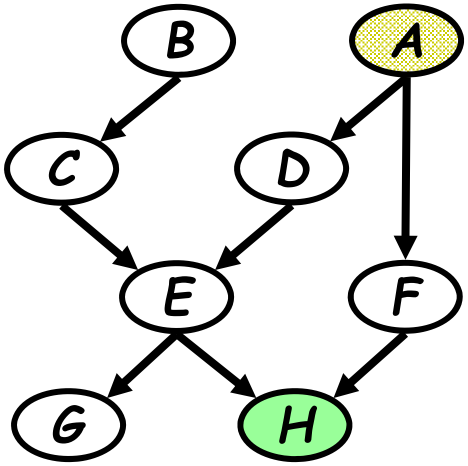


Example: Variable Elimination

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

- Initial factors:

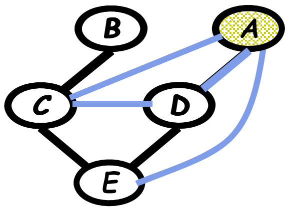
$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{P(e|c,d)m_f(a,e)}$$



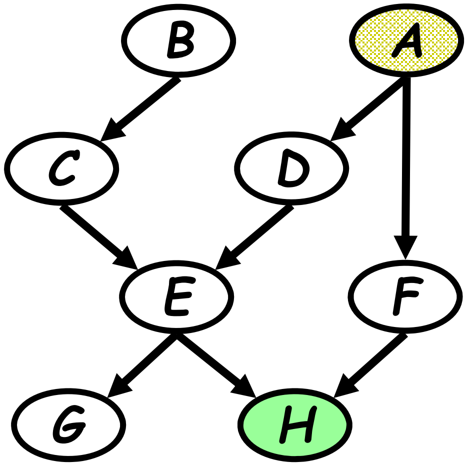
- Step 4: Eliminate E
  - compute

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

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



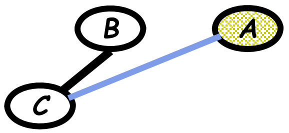
Example: Variable Elimination



- Query: $P(B \mid h)$
  - Need to eliminate: B, C, D
- Initial factors:

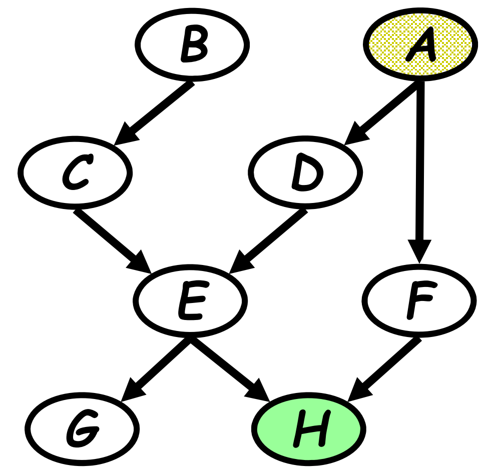
$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)m_f(a,e)$$
$$\Rightarrow P(a)P(b)P(c \mid b)\underline{P(d \mid a)m_e(a,c,d)}$$

- Step 5: Eliminate D
  - compute $$m_d(a,c) = \sum_d p(d \mid a)m_e(a,c,d)$$



$$\Rightarrow P(a)P(b)P(c \mid d)\underline{m_d(a,c)}$$

Example: Variable Elimination

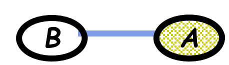


- Query: $P(B\,|h)$
  - Need to eliminate: B,C

- Initial factors:

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

- Step 6: Eliminate C

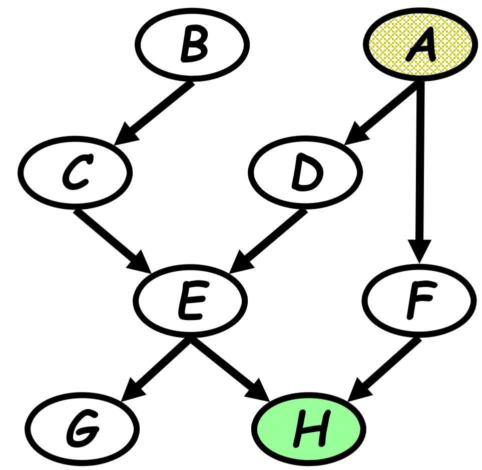


  - compute $$m_c(a,b)=\sum_c p(c\,|\,b)m_d(a,c)$$

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

Example: Variable Elimination

- Query: $P(B\,|h)$
  - Need to eliminate: B



- Initial factors:

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

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

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

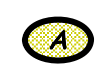
$$\Rightarrow P(a)P(b)P(c\,|\,d)P(d\,|\,a)P(e\,|\,c,d)m_f(a,e)$$

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

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

$$\Rightarrow P(a)\underline{P(b)m_c(a,b)}$$

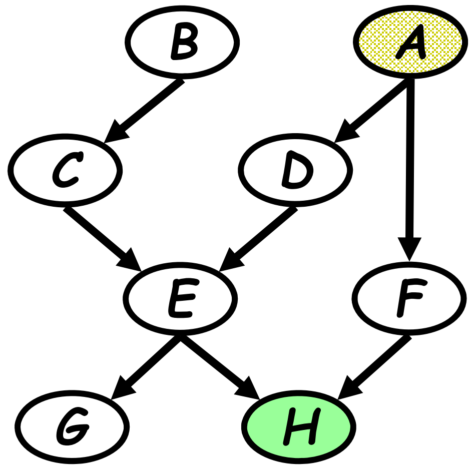
- Step 7: Eliminate B
  - compute $$m_b(a) = \sum_b p(b)m_c(a,b)$$



$$\Rightarrow P(a)\underline{m_b(a)}$$

Example: Variable Elimination

- Query: $P(B\,|h)$
  - Need to eliminate: B



- Initial factors:

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

- Step 8: Wrap-up

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

Complexity of variable elimination

- Suppose in one elimination step we compute

$$m_x(y_1,\ldots,y_k)=\sum_x m'_x(x,y_1,\ldots,y_k)$$

$$m'_x(x,y_1,\ldots,y_k)=\prod_{i=1}^{k} m_i(x,\mathbf{y}_{c_i})$$

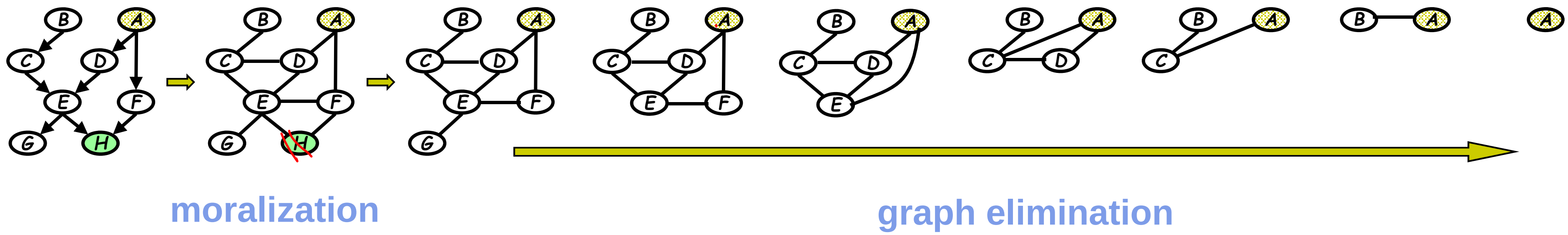
This requires

- $k \bullet |\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ multiplications
  - For each value for $x, y_1, \ldots, y_k$, we do k multiplications
- $|\mathrm{Val}(X)| \bullet \prod_i |\mathrm{Val}(\mathbf{Y}_{C_i})|$ additions
  - For each value of $y_1, \ldots, y_k$, we do $|Val(X)|$ additions

Complexity is exponential in number of variables in the intermediate factor

Understanding Variable Elimination

- A graph elimination algorithm

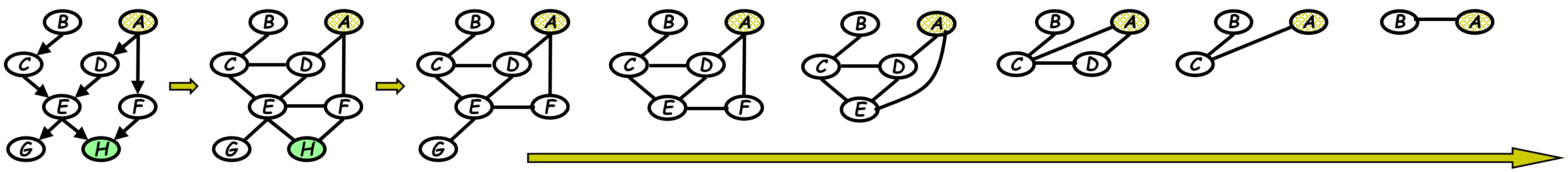


Graph elimination

- 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

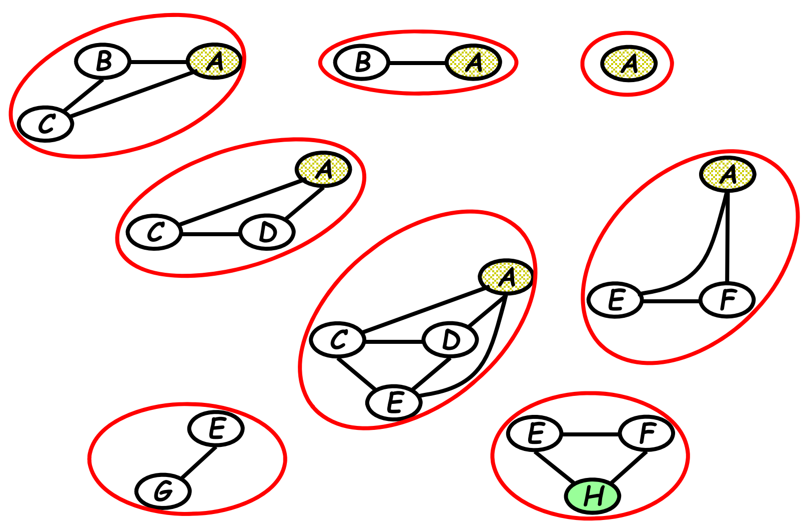
Understanding Variable Elimination

- A graph elimination algorithm

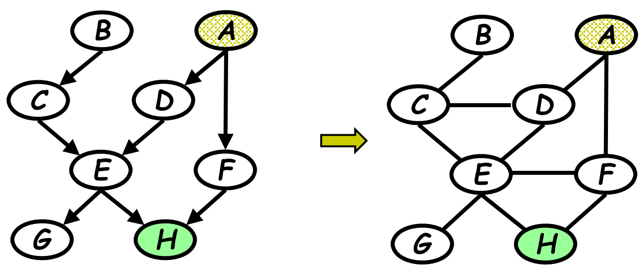


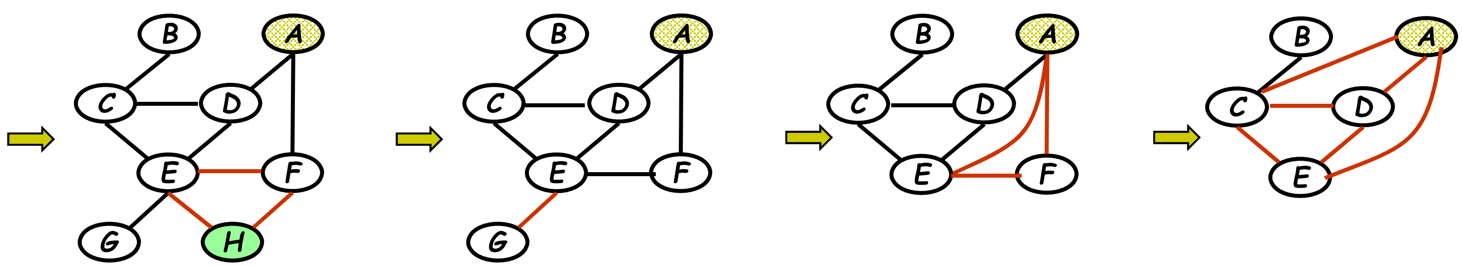
moralization   graph elimination

- Intermediate terms correspond to the cliques resulted from elimination

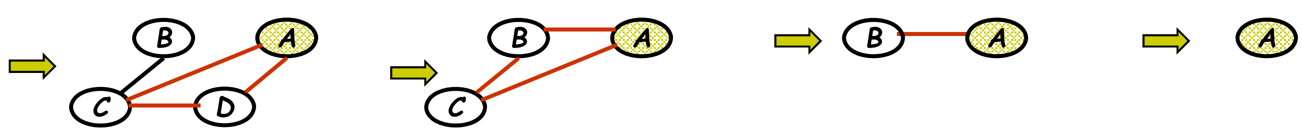


Elimination Cliques





$m_h(e, f)$ $\quad$ $m_g(e)$ $\quad$ $m_f(e,a)$ $\quad$ $m_e(a,c,d)$

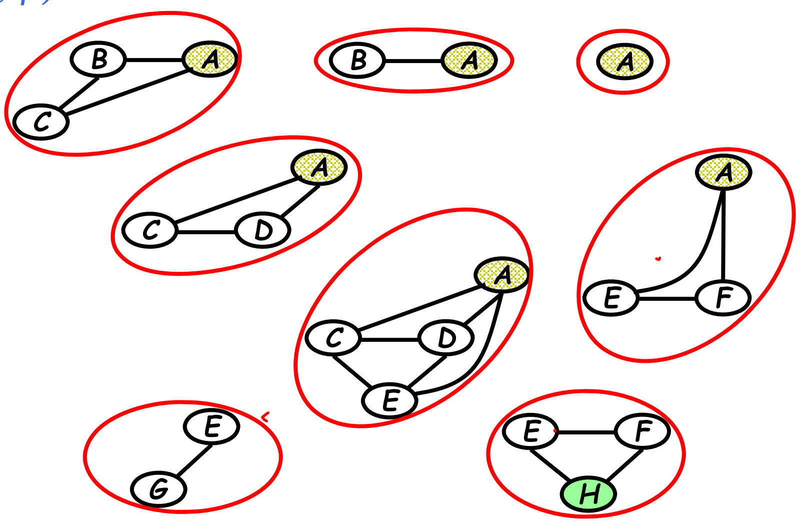


$m_d(a,c)$ $\quad$ $m_c(a,b)$ $\quad$ $m_b(a)$

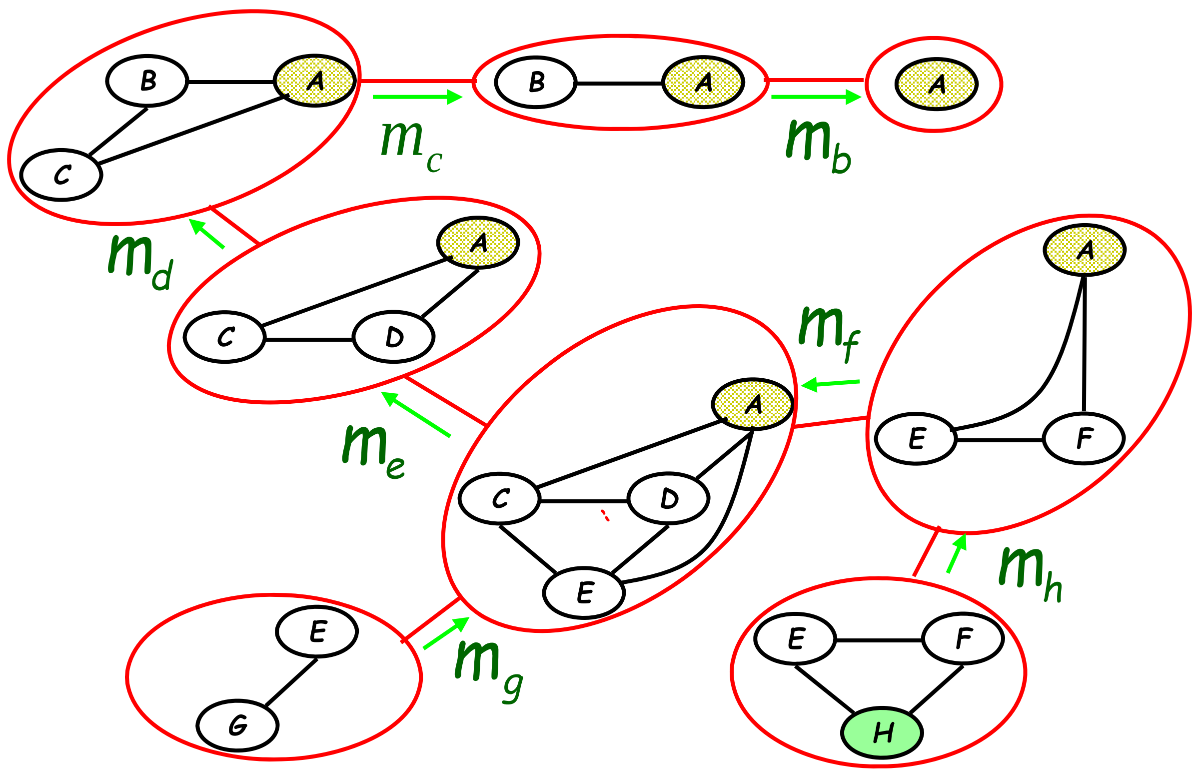
Graph elimination and marginalization

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

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



A clique tree



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

Complexity

- 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

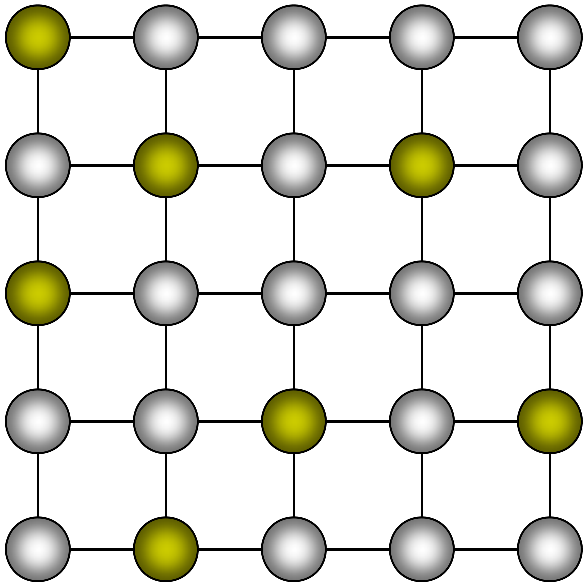
  → Inference is NP-hard

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

Examples

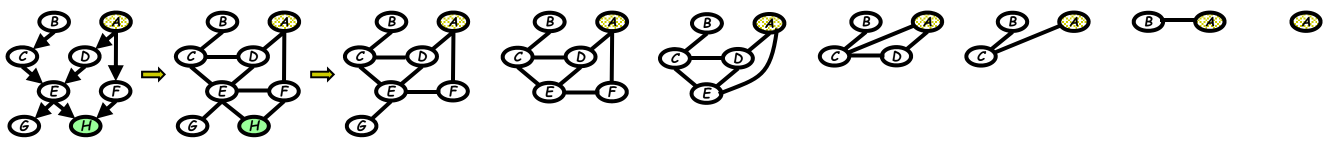
- Star
- Tree

More example: Ising model



Limitation of Procedure Elimination

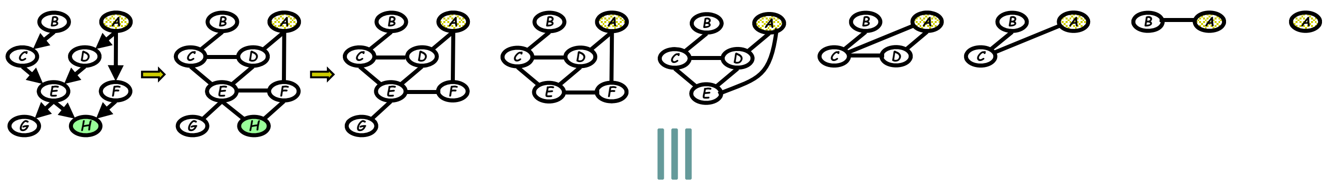
- Limitation

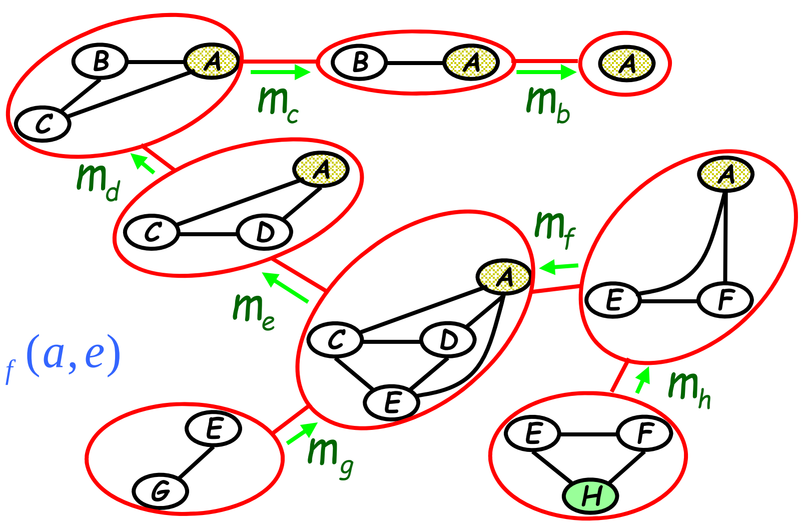


From Elimination to Message Passing

- 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**



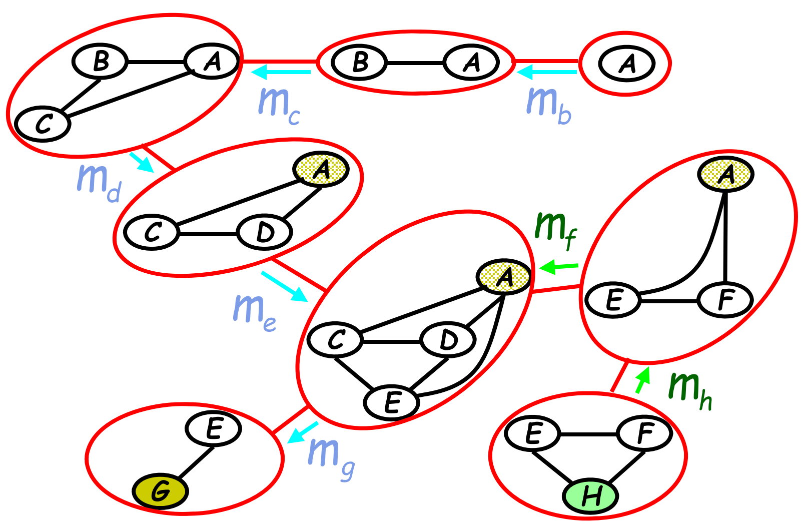


$$m_e(a,c,d) = \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

- 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 $\equiv$ 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.