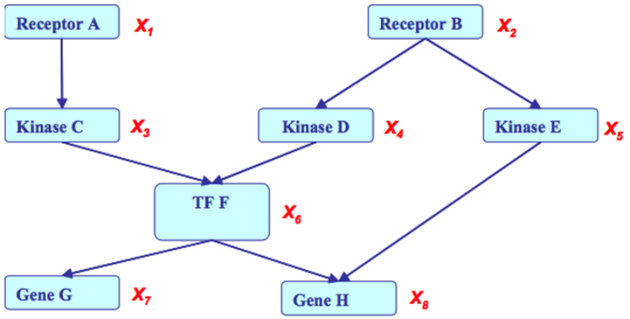
# Machine Learning

## Lecture 11

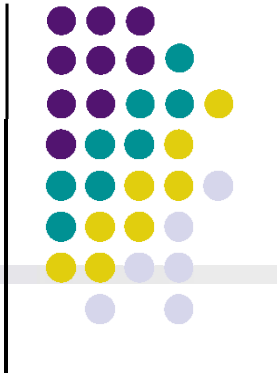
Yang Yang
Department of Computer Science & Engineering
Shanghai Jiao Tong University

# Graphical Models




**Reading: Chap. 8, C.B book**

# Two Types of GMs



- Directed edges give causality relationships ( Bayesian Network or Directed Graphical Model ):

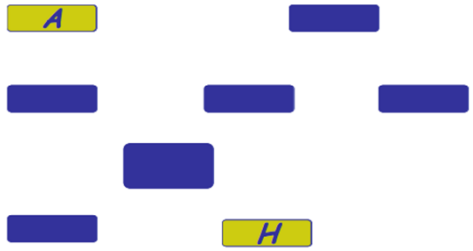
- Undirected edges simply give correlations between variables ( Markov Random Field or Undirected Graphical model ):

# Recap of Basic Prob. Concepts

- Representation: what is the joint probability dist. on multiple variables ?

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

  - How many state configurations in total ? — $2^8$
  - Are they all needed to be represented ?
  - Do we get any scientific / medical insight ?

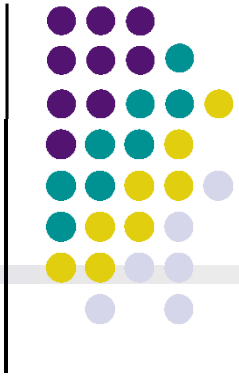



- Learning: where do we get all this probabilities ?
  - Maximal-likelihood estimation ? but how many data do we need ?
  - Where do we put domain knowledge in terms of plausible relationships between variables, and plausible values of the probabilities ?

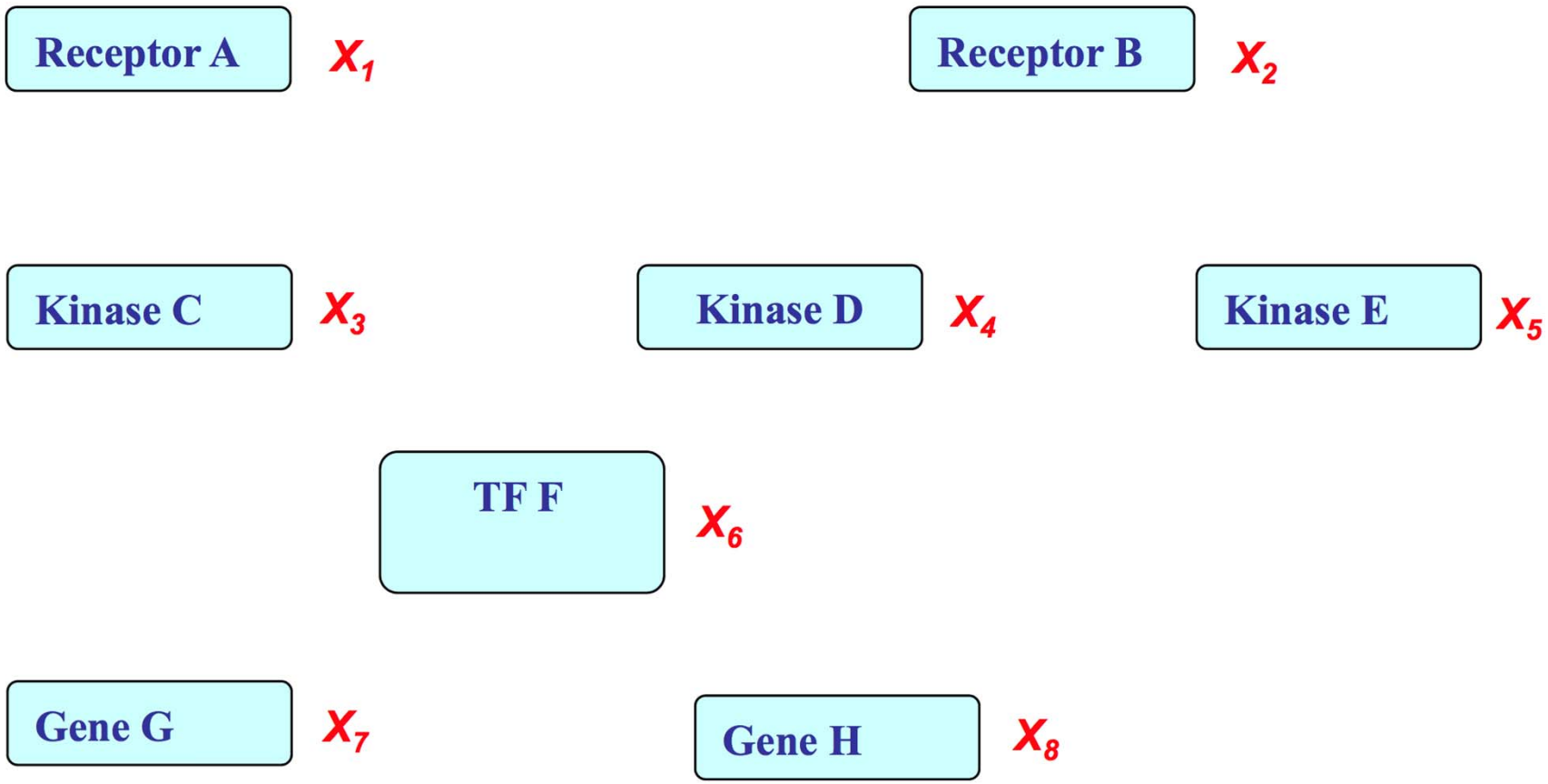
- Inference: If not all variables are observable, how to compute the conditional distribution of latent variables given evidence ?
  - Computing p(H|A) would require summing over all $2^6$ configurations of the unobserved variables

# What is a Bayesian Network ?

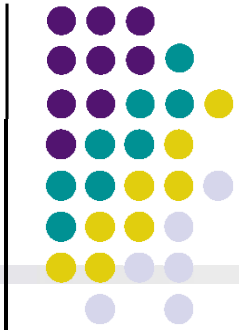
— example from a single transduction pathway



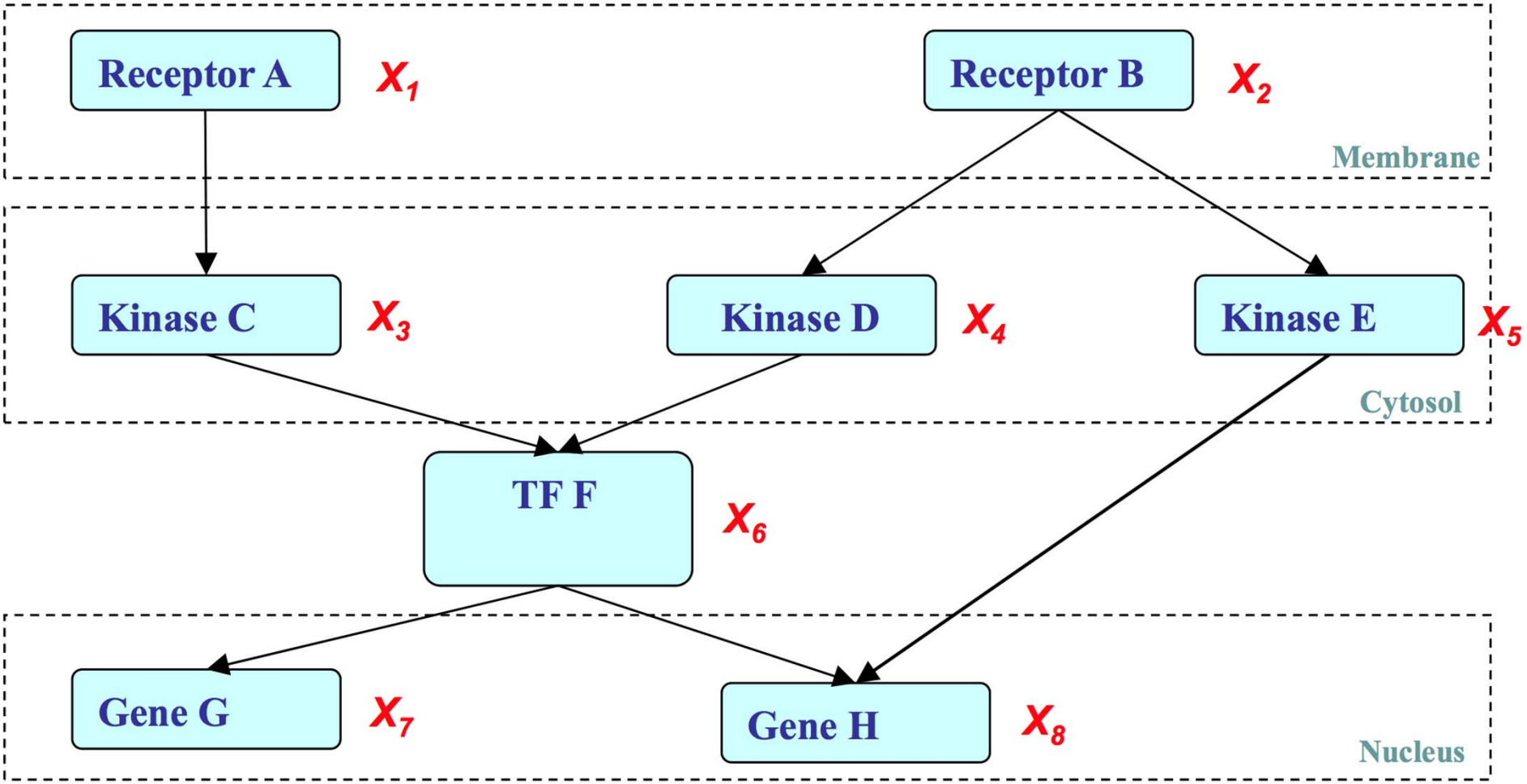
- A possible world for cellular signal transduction:



# BN: Structure Simplifies Representation

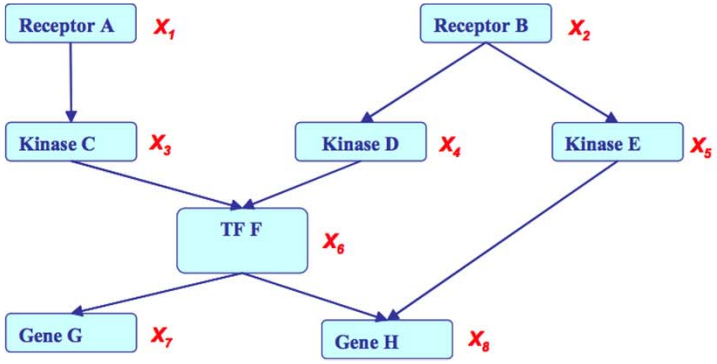


- Dependencies among variables



# Bayesian Network

- If $X_i$'s are conditionally independent (as described by a BN), the joint can be factored to a product of simpler terms, e.g.,



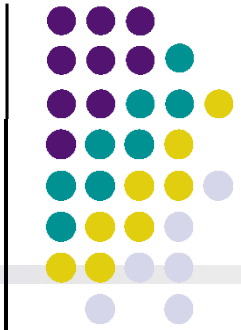

$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

$$= P(X_1)\, P(X_2)\, P(X_3|\, X_1)\, P(X_4|\, X_2)\, P(X_5|\, X_2)$$
$$P(X_6|\, X_3, X_4)\, P(X_7|\, X_6)\, P(X_8|\, X_5, X_6)$$

- Why we may favor a BN ?
  - Representation cost: how many probability statements are needed ?

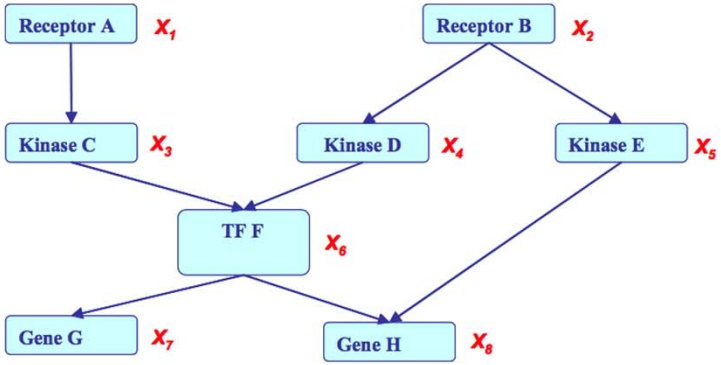
    2+2+4+4+4+8+4+8=36, an 8-fold reduction from $2^8$ !
  - Algorithms for systematic and efficient inference / learning computation
    - Exploring the graph structure and probabilistic semantics
  - Incorporation of domain knowledge and causal (logical) structures

# Bayesian Network



- A BN is a directed graph whose nodes represent the random variables and whose edges represent direct influence of one variable on another.
- It is a data structure that provides the skeleton for representing a joint distribution compactly in a factorized way.
- It offers a compact representation for a set of conditional independence assumptions about a distribution.
- We can view the graph as encoding a generative sampling process executed by nature, where the value for each variable is selected by nature using a distribution that depends only on its parents. In other words, each variable is a stochastic function of its parents.

# Bayesian Network: Factorization Theorem




$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$

$$= P(X_1)\, P(X_2)\, P(X_3|\, X_1)\, P(X_4|\, X_2)\, P(X_5|\, X_2)$$
$$P(X_6|\, X_3, X_4)\, P(X_7|\, X_6)\, P(X_8|\, X_5, X_6)$$

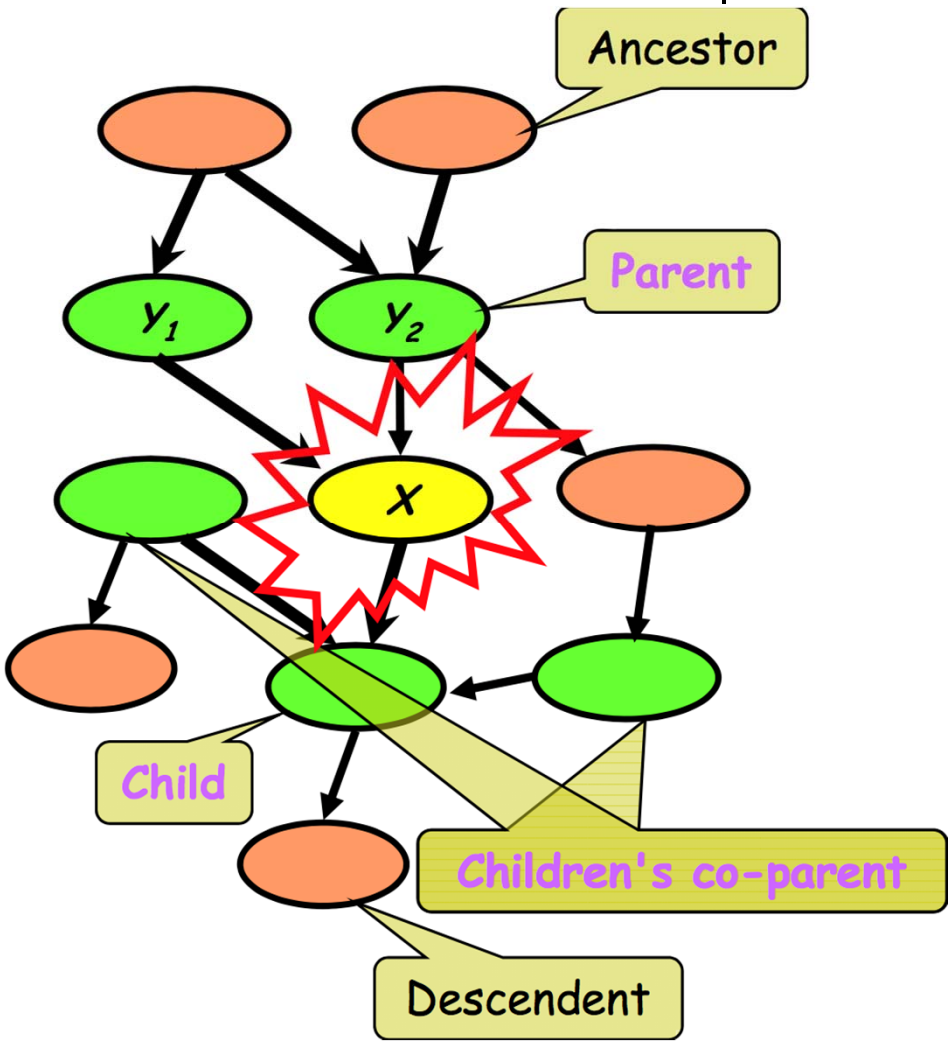
- Theorem:
  Given a DAG, the most general form of the probability distribution that is consistent with the (probabilistic independence properties encoded in the) graph factors according to "node given its parents":

$$P(\mathbf{X}) = \prod_{i=1:d} P(X_i \mid \mathbf{X}_{\pi_i})$$

where $X_{\Pi i}$ is the set of parents of xi. d is the number of nodes (variables) in the graph.

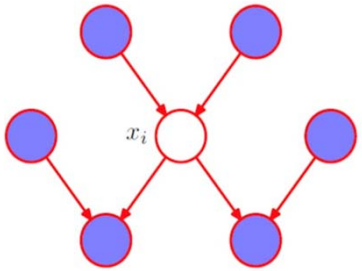
# **Bayesian Network:** Conditional Independence Semantics

- Structure: **DAG**
  - Meaning: a node is conditionally independent of every other node in the network outside its Markov blanket
  - Local conditional distributions (CPD) and the DAG completely determine the joint dist.
  - Give causality relationships, and facilitate a generative process



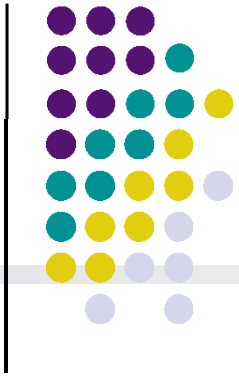
# Markov Blanket

$$p(x_i|x_{\{j\neq i\}}) = \frac{p(x_1,...,x_D)}{\int p(x_1,...,x_D)dx_i}$$
$$= \frac{\prod_k p(x_k|pa_k)}{\int \prod_k p(x_k|pa_k)dx_i}$$



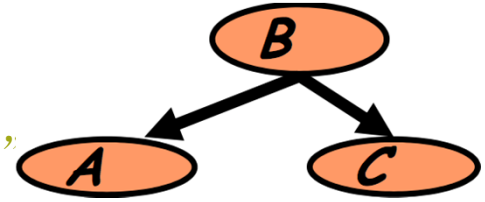

- The Markov blanket of a node xi comprises the set of parents, children and co-parents of the node
- The conditional distribution of xi, conditioned on all the remaining variables in the graph, is dependent only on the variables in the Markov blanket.
- The Markov blanket of a node xi as being the minimal set of nodes that isolates xi from the rest of the graph.

# Local Structures & Independencies



- Common parent
  - Fixing B decouples A and C

    "given the level of gene B, the levels of A and C are independent"



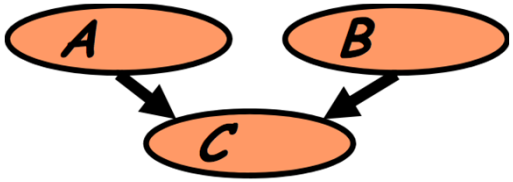

- Cascade
  - Knowing B decouples A and C

    "given the level of gene B, the level gene A provides no extra prediction value for the level of gene C"


- V-structure
  - Knowing C couples A and B

    because A can "explain away" B w.r.t. C

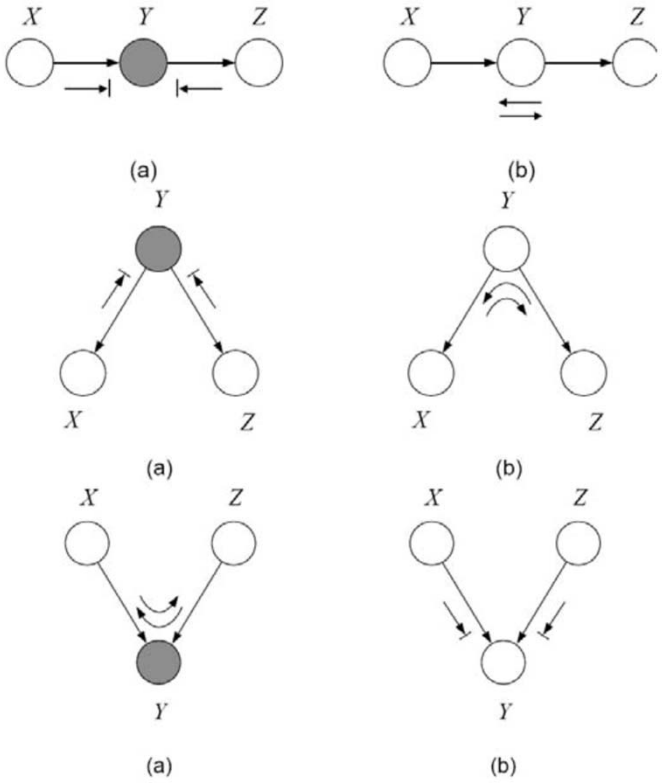
    "If A correlates to C, then chance for B to also correlate to B will decrease"




- The language is compact, the concepts are rich !

# Global Markov Properties of DAGs

- X is d-separated (directed-separated) from Z given Y if we can't send a ball from any node in X to any node in Z using the "Bayes-ball" algorithm illustrated below (and plus some boundary conditions):

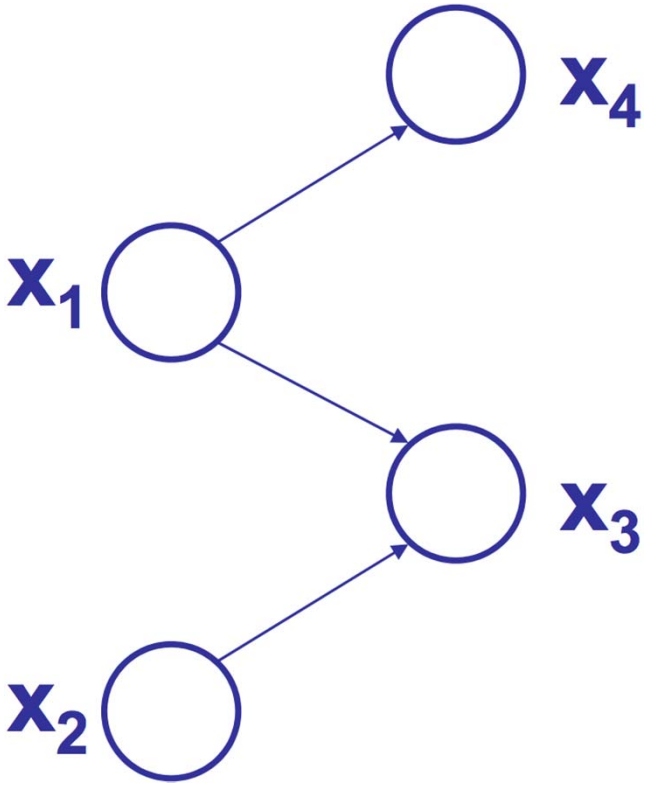



- Defn: I(G) = all independence properties that correspond to d-separation:

$$I(G) = \{X \perp Z | Y : dsep_G(X; Z|Y)\}$$

- D-separation is sound and complete

# Example:

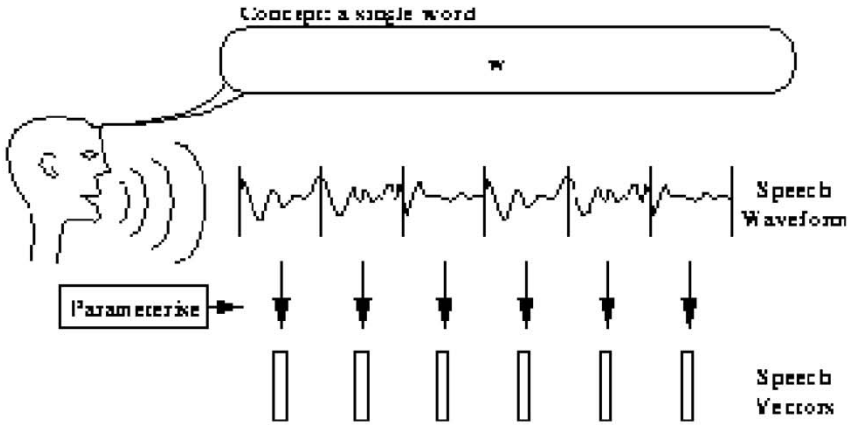


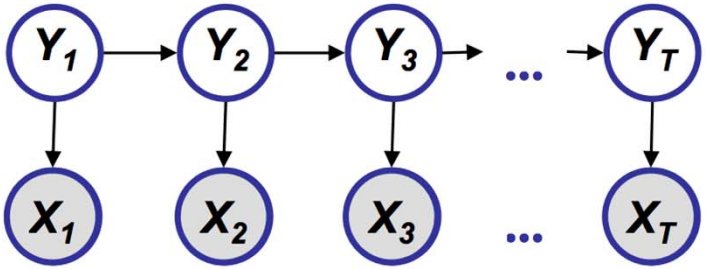

- Complete the I(G) of this graph:

**Essentially: A BN is a database of Pr. Independence statements among variables.**

# Example

- Speech recognition

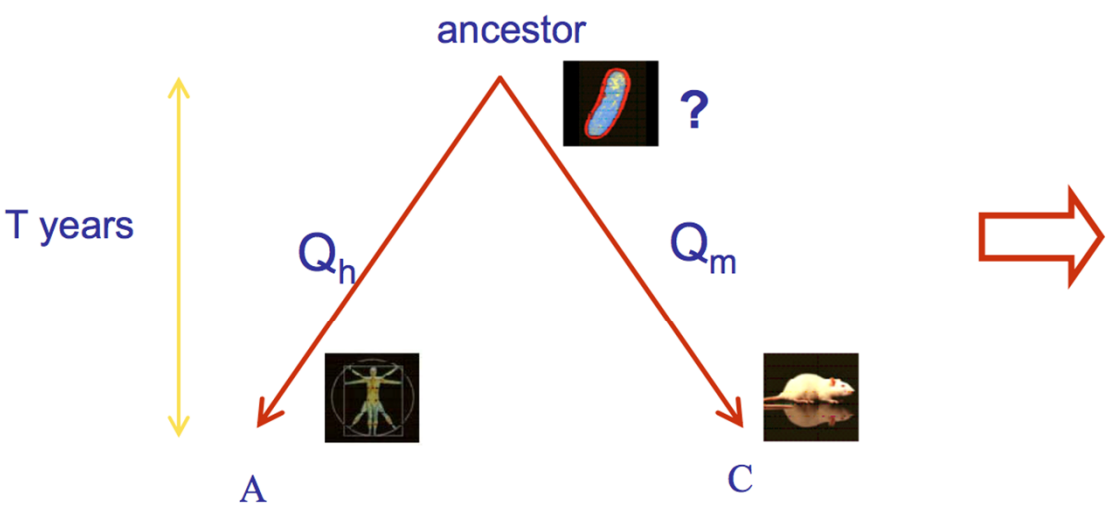


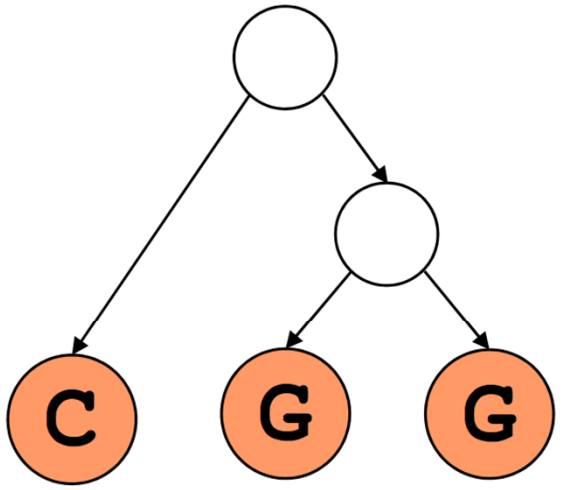




**Hidden Markov Model**

# Example, cont.

- Evolution

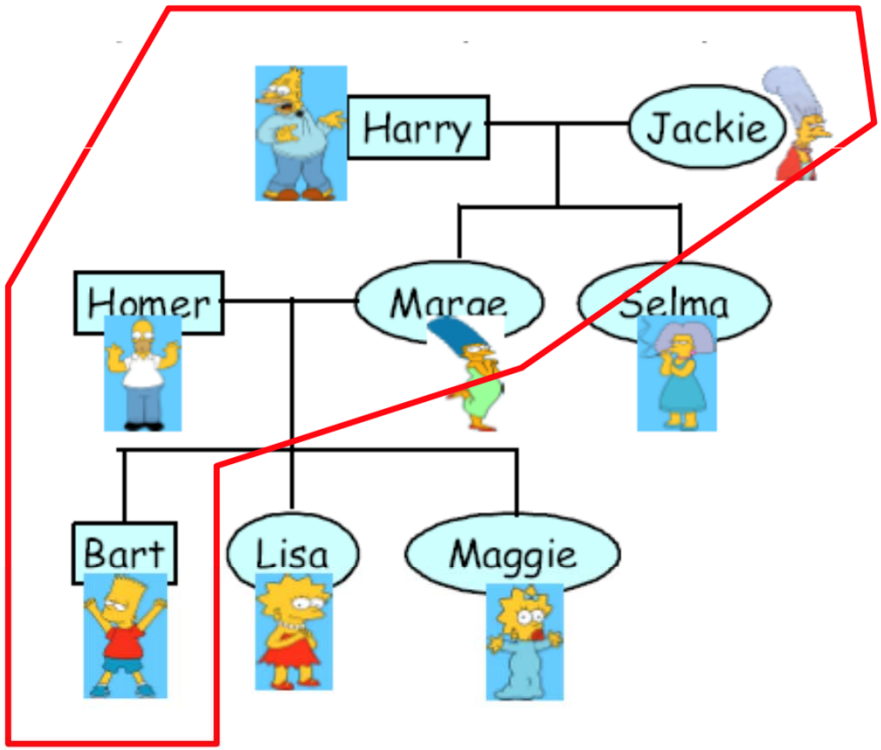


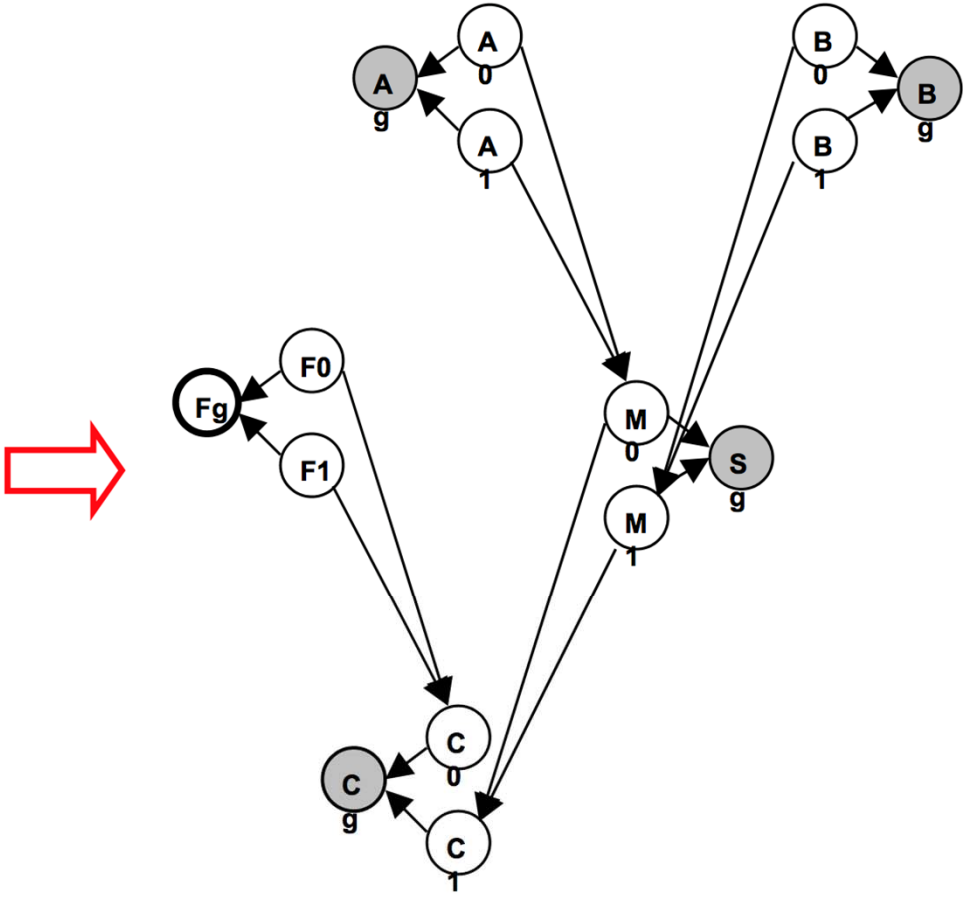




**Tree Model**

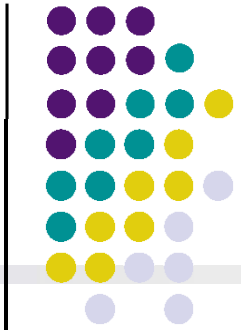
# Example: a pedigree of people

- Genetic Pedigree



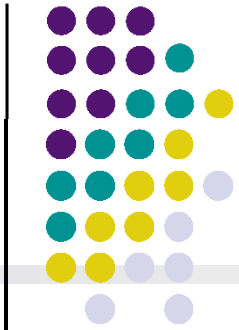
# Towards Quantitative Specification of Probability Distribution



- Separation properties in the graph imply independence properties about the associated variables
- For the graph to be useful, any conditional independence properties we can derive from the graph should hold for the probability distribution that the graph represents
- **The Equivalence Theorem**

  For a graph G,
  Let D1 denote the family of all distributions that satisfy I(G),
  Let D2 denote the family of all distributions that factor according to G,
  Then D1 ≡ D2.

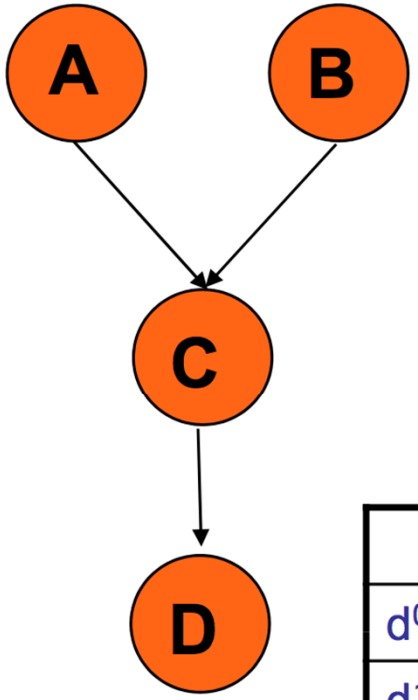
# Conditional Probability Tables (CPTs)



| | |
|---|---|
| $a^0$ | 0.75 |
| $a^1$ | 0.25 |

| | |
|---|---|
| $b^0$ | 0.33 |
| $b^1$ | 0.67 |

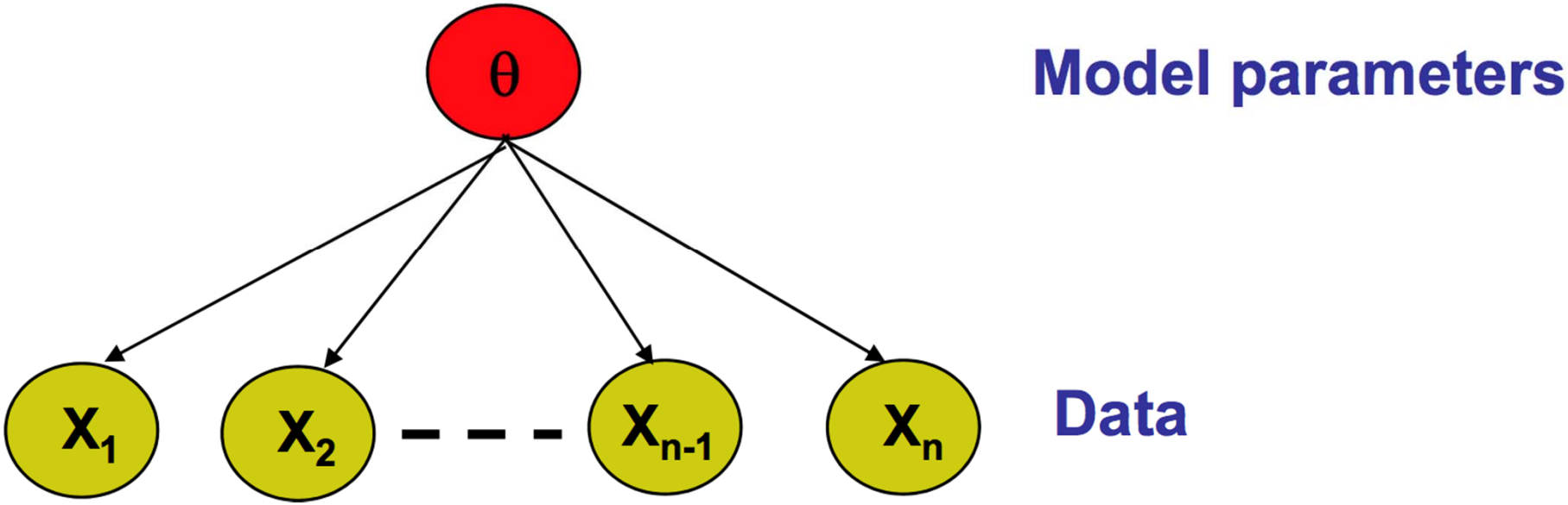
$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



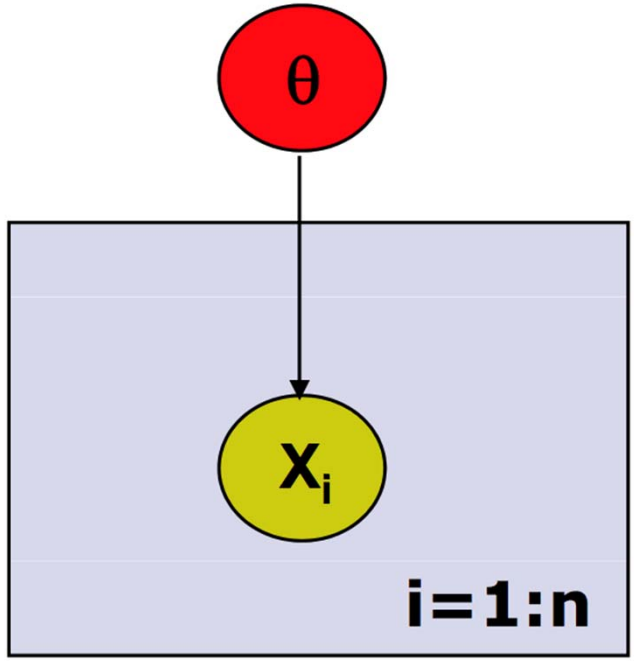

| | $a^0b^0$ | $a^0b^1$ | $a^1b^0$ | $a^1b^1$ |
|---|---|---|---|---|
| $c^0$ | 0.45 | 1 | 0.9 | 0.7 |
| $c^1$ | 0.55 | 0 | 0.1 | 0.3 |

| | $c^0$ | $c^1$ |
|---|---|---|
| $d^0$ | 0.3 | 0.5 |
| $d^1$ | 07 | 0.5 |

# Conditionally Independent Observations



# "Plate" Notation



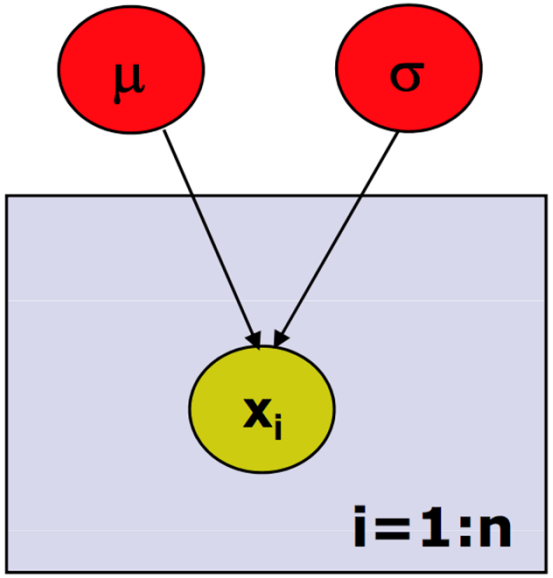

Model parameters

Data = $\{x_1,\ldots,x_n\}$

Plate = rectangle in graphical model

variables within a plate are replicated
in a conditionally independent manner

# Example: Gaussian Model



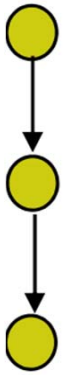

Generative model:

$P(x_1,\ldots,x_n \mid \mu, \sigma) = P\ p(x_i \mid \mu, \sigma)$
$= p(\text{data} \mid \text{parameters})$
$= p(D \mid \theta)$
where $\theta = \{\mu, \sigma\}$

- Likelihood = p(data | parameters)
  = p(D | θ)
  = L(θ)
- Likelihood tells us how likely the observed data are conditioned on a particular setting of the parameters
  - Often easier to work with log L(θ)

# Summary

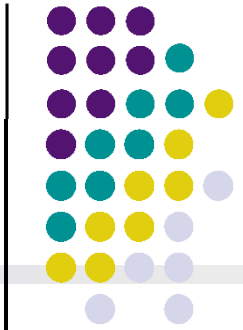
- Represent dependency structure with a directed acyclic graph
  - Node <-> random variable
  - Edges encode dependencies
    - Absence of edge -> conditional independence
  - Plate representation
  - A GM is a database of prob. independence statement on variables



- The factorization theorem of the joint probability
  - Local specification -> globally consistent distribution
  - Local representation for exponentially complex state-space
  - It is a smart way to write/specify/compose/design exponentially-large probability distribution without paying an exponential cost, and at the same time endow the distributions with structured semantics

- Support efficient inference and learning

# Inference and Learning



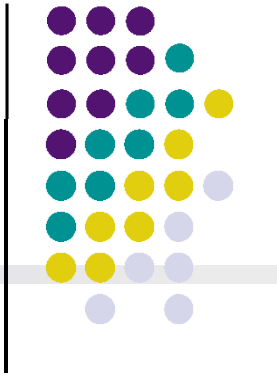
- We now have compact representations of probability distribution: GM
- A BN M describes a unique probability distribution P
- Typical tasks:
  - Task 1: How do we answer queries about P ?
    - We use inference as a name for the process of computing answers to such queries
  - Task 2: How do we estimate 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*

# Inferential Query 1: Likelihood



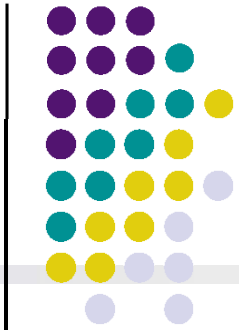
- Most of the queries one may ask involve evidence
  - Evidence $x_v$ is an assignment of values to a set $\mathbf{X_v}$ of nodes in the GM over variable set $\mathbf{X}=\{X_1, X_2, \ldots, X_n\}$
  - Without loss of generality $\mathbf{X_v}=\{X_{k+1}, \ldots, X_n\}$
  - Write $\mathbf{X_H}=\mathbf{X}\backslash\mathbf{X_V}$ as the set of hidden variables, $\mathbf{X_H}$ can be $\emptyset$ or $\mathbf{X}$

- Simplest query: compute probability of evidence

$$P(x_v) = \sum_{x_H} P(X_H, X_v) = \sum_{x_1} \ldots \sum_{x_k} P(x_1, \ldots, x_k, x_v)$$

  - this is often referred to as computing the likelihood of $x_v$

# Inferential Query 2: Conditional Probability



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

$$P(X_H|X_V = x_V) = \frac{P(X_H, x_V)}{P(x_V)} = \frac{P(X_H, x_V)}{\sum_{x_H} P(X_H = x_H, x_V)}$$

  - this is the a *posteriori* belief in $X_H$, given evidence $x_v$

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

$$P(Y|x_V) = \sum_z P(Y, Z = z|x_V)$$

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

# Applications of a *posteriori* Belief

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

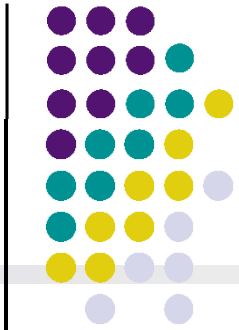

  - the query node is a descendent of the evidence

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

# Inferential Query 3: Most Probable Assignment



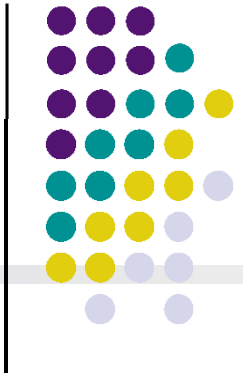
- In this query we want to find the most probable joint assignment (MPA) for some variables of interest

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

$$Y^*|x_V = argmax_y P(Y|x_V) = argmax_y \sum_z P(Y, Z = z|x_V)$$

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

# Complexity of Inference

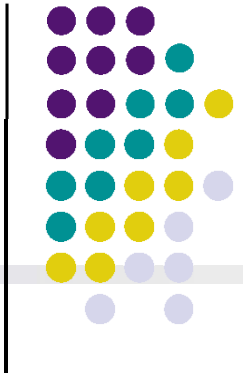


**Thm:**

**Computing $P(X_H=x_H|x_v)$ in an arbitrary GM is NP-hard**

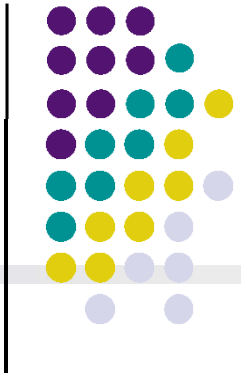
- Hardness does not mean we cannot solve inference
  - It implies that we cannot find a general procedure that works efficiently for arbitrary GMs
  - For particular families of GMs, we can have provably efficient procedures

# Approaches to Inference



- Exact inference algorithms
  - The elimination algorithm
  - Belief propagation
  - The junction tree algorithms (but will not cover in detail here)
- Approximate inference techniques
  - Variational algorithms
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods

# Inference on General BN via Variable Elimination



**General idea:**

- Write query in the form

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

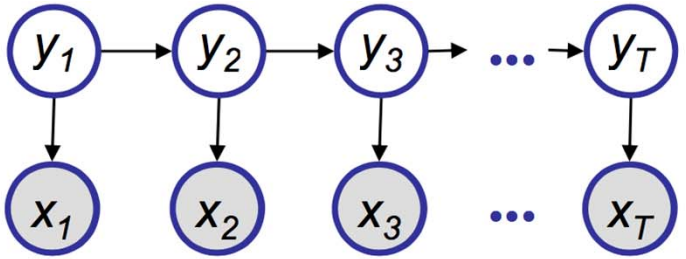
  - this suggests an "elimination order" of latent variables to be marginalized

- 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

$$P(X_1 | e) = \frac{P(X_1, e)}{P(e)}$$

# Hidden Markov Model




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

These are known as "forward message"

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

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

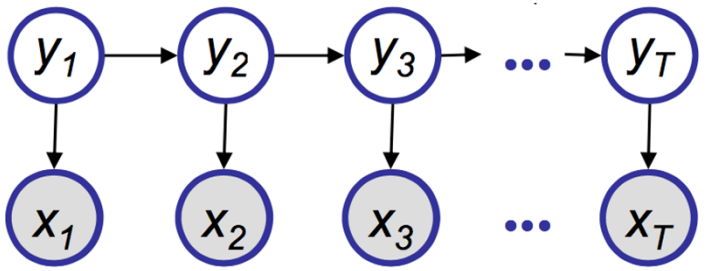
$$= \sum_{y_2} \cdots \sum_{y_T} P(x_2|y_2) P(y_3|y_2) \cdots \sum_{y_1} P(y_1) P(x_1|y_1) P(y_2|y_1)$$

$$\searrow P(x_1, y_2)$$

$$= \sum_{y_3} \cdots \sum_{y_T} P(x_3|y_3) P(y_4|y_3) \cdots \sum_{y_2} P(x_1, y_2) P(x_2|y_2) P(y_3|y_2)$$

$$\searrow P(x_1, x_2, y_3)$$

# Hidden Markov Model




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

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

$$= \sum_{y_1} \cdots \sum_{y_{T-1}} P(y_1) \cdots P(x_{T-1}|y_{T-1}) \sum_{y_T} P(y_T|y_{T-1}) P(x_T|y_T)$$

$$= \sum_{y_1} \cdots \sum_{y_{T-2}} P(y_1) \cdots P(x_{T-2}|y_{T-2}) \sum_{y_{T-1}} P(y_{T-1}|y_{T-2}) P(x_{T-1}|y_{T-1}) P(x_T|y_{T-1})$$

$$\rightarrow P(x_T|y_{T-1})$$

$$\rightarrow P(x_T, x_{T-1}|y_{T-2})$$

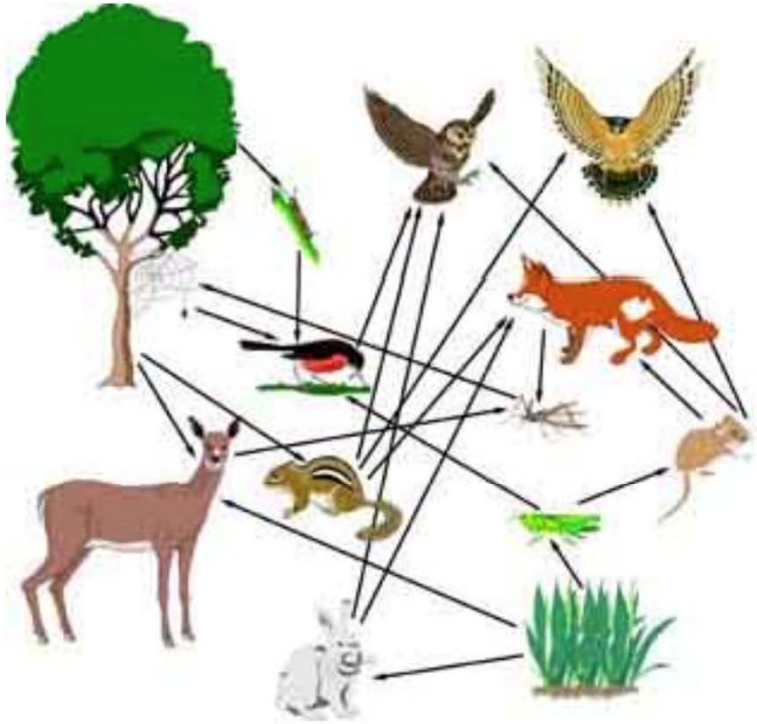
$$= \cdots$$

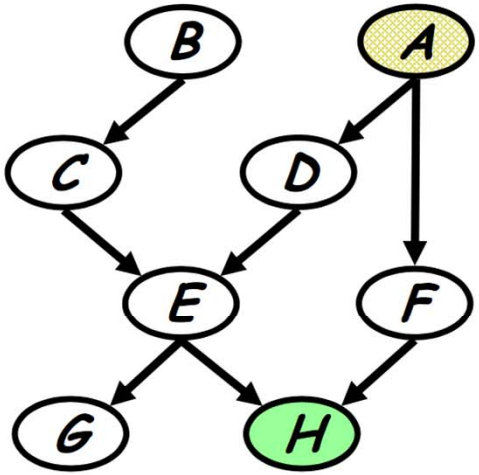
$$= \cdots$$

These are known as "backward message"

# A Bayesian Network

## A food web





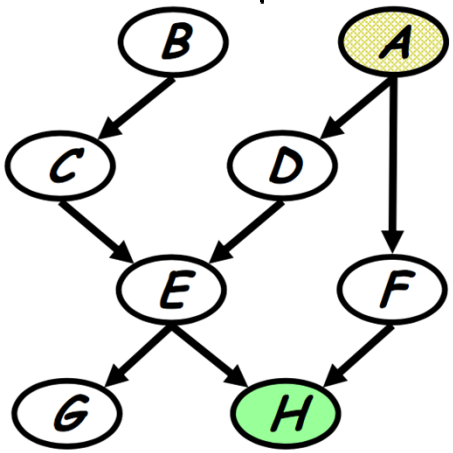

What is the probability that hawks are leaving given that the grass condition is poor?

# Example: Variable Elimination

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

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

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

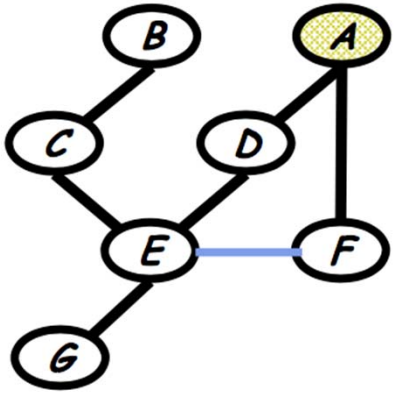



- Step 1:
  - Conditioning (fix the evidence node (i.e., h) on its observed value (i.e., $\tilde{h}$)):

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

  - This step is isomorphic to a marginalization step:

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

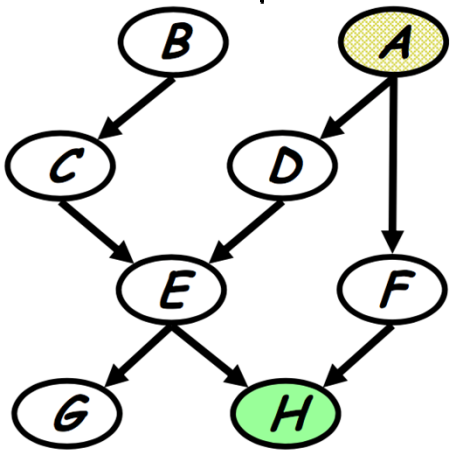


# Example: Variable Elimination

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

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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

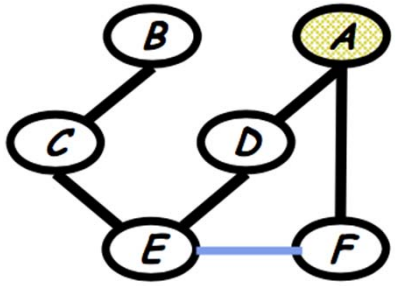



- Step 2: Eliminate G
  - compute

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

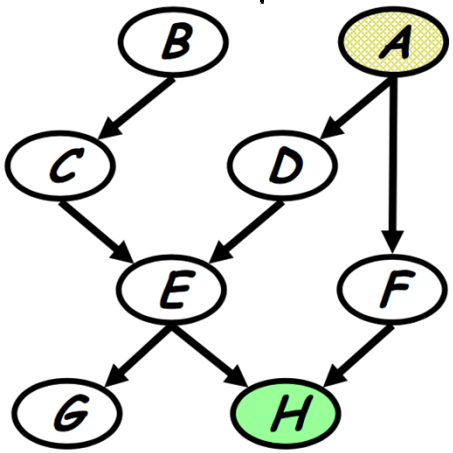
$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_g(e)m_h(e,f)$$

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



# Example: Variable Elimination

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




- Initial factors:

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

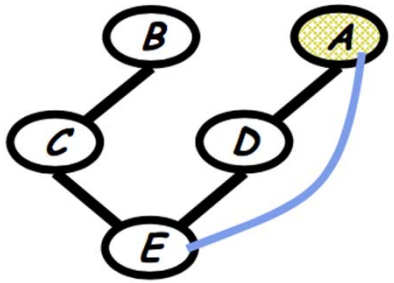
$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{P(f|a)m_h(e,f)}$$

- Step 3: Eliminate F
  - compute

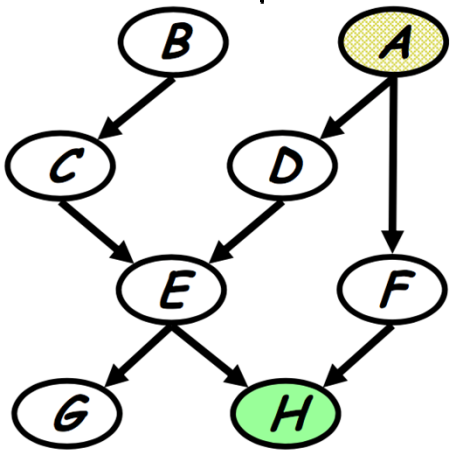
$$m_f(e,a) = \sum_f p(f|a)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)\underline{m_f(a,e)}$$



# Example: Variable Elimination

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




- Initial factors:

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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

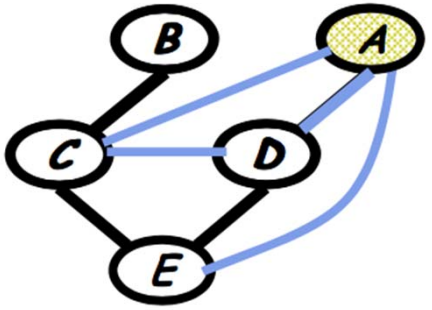
$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)\underline{P(e|c,d)m_f(a,e)}$$

- Step 4: Eliminate E
  - compute

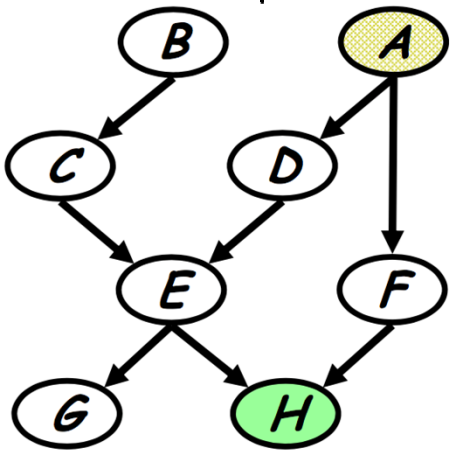
$$m_e(a,c,d) = \sum_e p(e|c,d)m_f(a,e)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)\underline{m_e(a,c,d)}$$



# Example: Variable Elimination

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




- Initial factors:

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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

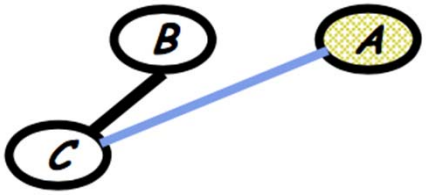
$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$

$$=> \; P(a)P(b)P(c|b)\underline{P(d|a)m_e(a,c,d)}$$

- Step 5: Eliminate D
  - compute

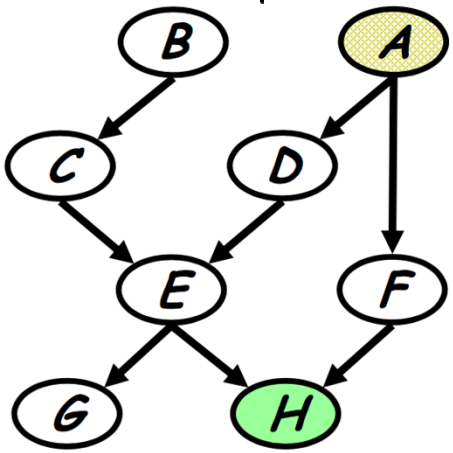
$$m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$$

$$=> \; P(a)P(b)P(c|b)\underline{m_d(a,c)}$$



# Example: Variable Elimination

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




- Initial factors:

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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)m_e(a,c,d)$$

$$=> \; P(a)P(b)P(c|b)m_d(a,c)$$

- Step 6: Eliminate C
  - compute

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


$$=> \; P(a)P(b)m_c(a,b)$$

# Example: Variable Elimination

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

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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

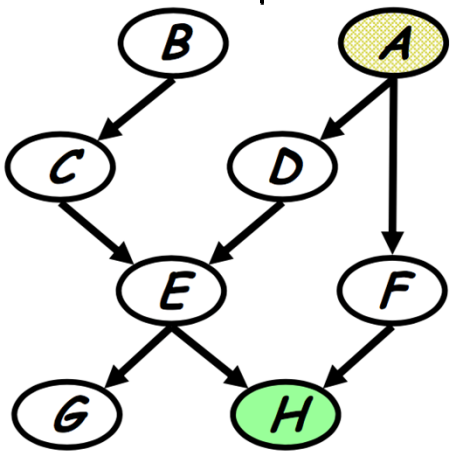
$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)m_e(a,c,d)$$

$$=> \; P(a)P(b)P(c|b)m_d(a,c)$$

$$=> \; P(a)\underline{P(b)m_c(a,b)}$$



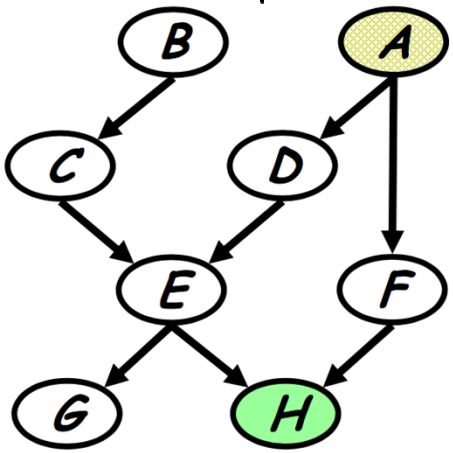

- Step 7: Eliminate B
  - compute

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

$$=> \; P(a)\underline{m_b(a)}$$

# Example: Variable Elimination

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




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

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)m_h(e,f)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)P(e|c,d)m_f(a,e)$$

$$=> \; P(a)P(b)P(c|b)P(d|a)m_e(a,c,d)$$

$$=> \; P(a)P(b)P(c|b)m_d(a,c)$$

$$=> \; P(a)P(b)m_c(a,b)$$

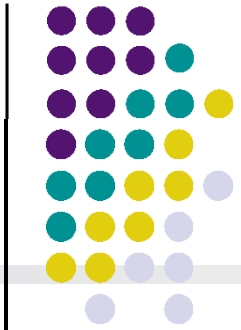
$$=> \; P(a)m_b(a)$$

- Step 8: Wrap-up

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

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

# Complexity of Variable Elimination



- Suppose in one elimination step we compute

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

$$m'_x(x, y_1, ..., y_k) = \prod_{i=1}^{k} m_i(x, y_{c_i})$$

- This requires

- $k \cdot |Val(X)| \cdot \prod_i |Val(y_{c_i})|$ multiplications

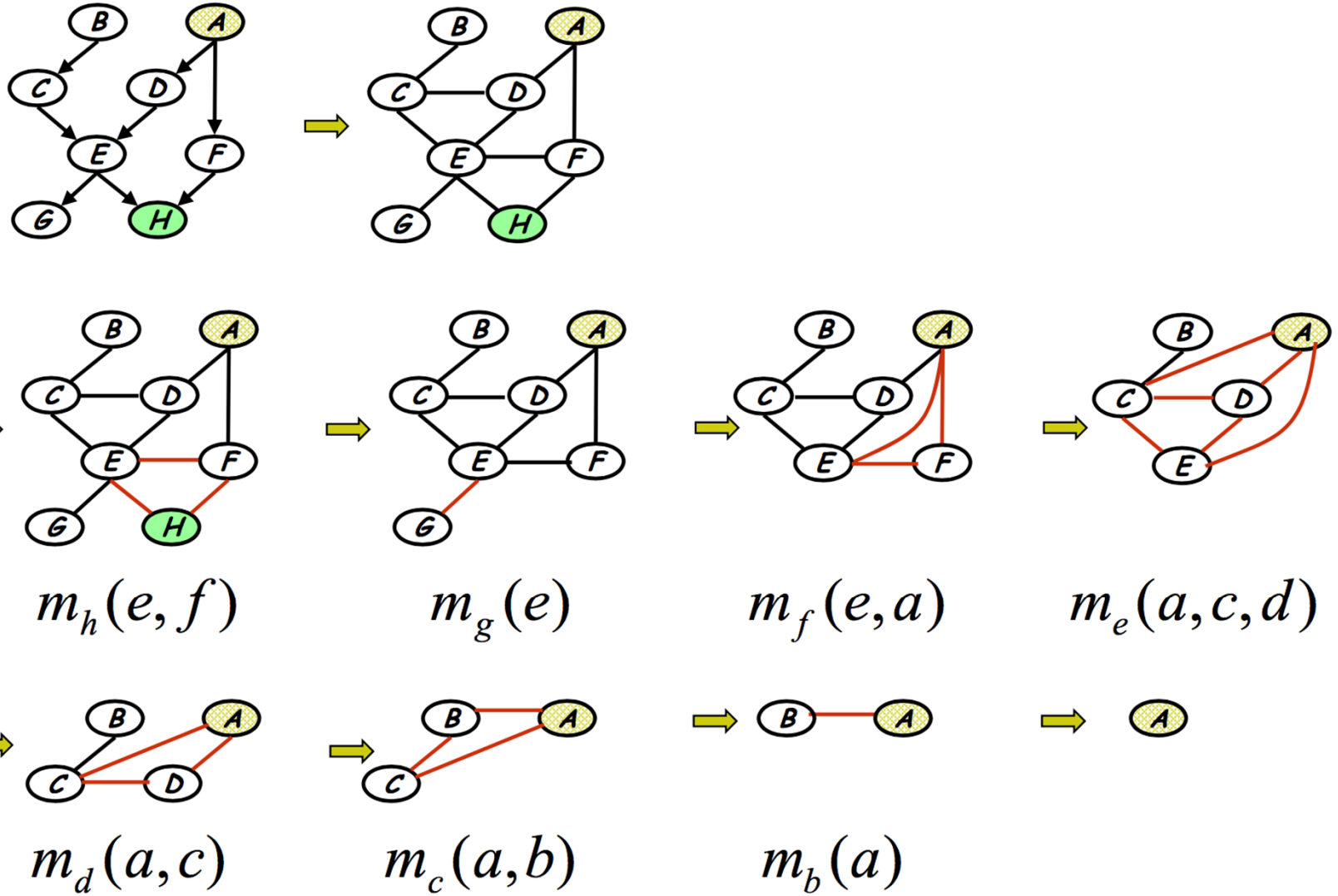
  — For each value of x, $y_1$, …, $y_k$, we do k multiplications

- $|Val(X)| \cdot \prod_i |Val(y_{c_i})|$ additions

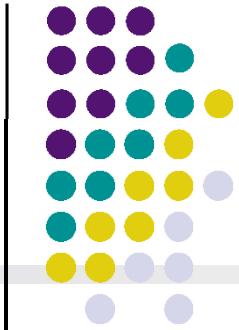
  — For each value of $y_1$, …, $y_k$, we do |*Val(X)*| additions

Complexity is exponential in number of variables in the intermediate factor

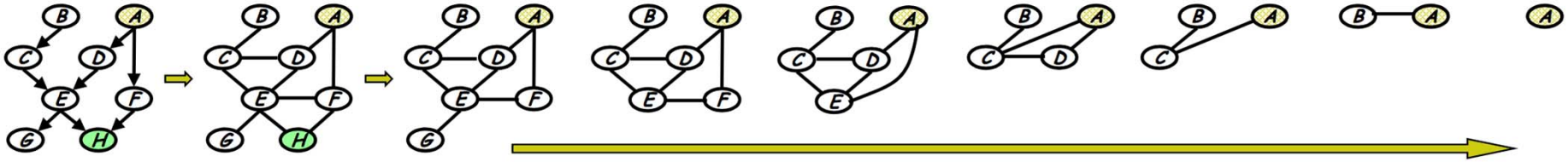
# Elimination Cliques



# Understanding Variable Elimination



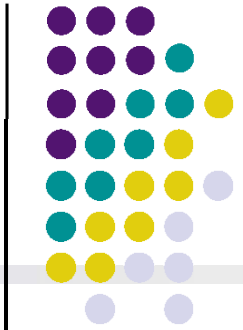
- A graph elimination algorithm




- Intermediate terms correspond to the cliques resulted from elimination
  - "good" elimination orderings lead to small cliques and hence reduce complexity
    (what will happen if we eliminate "e" first in the above graph ?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found

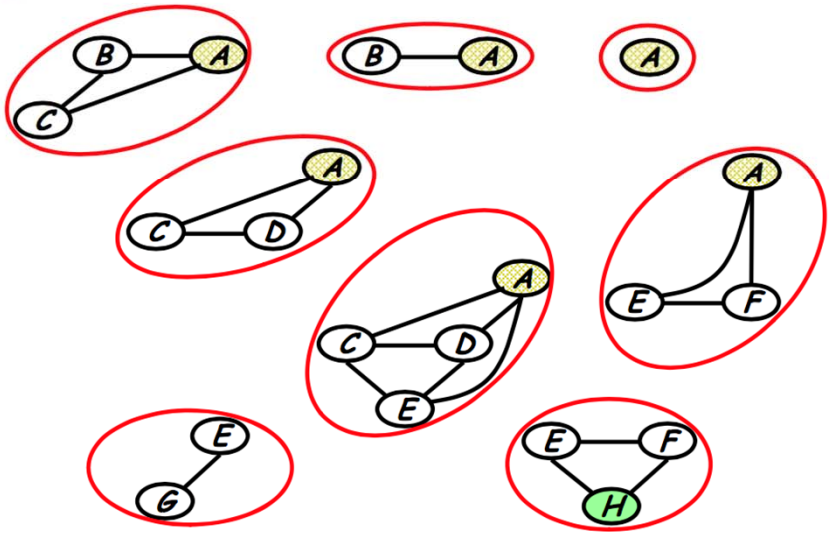
- Applies to undirected GMs

# From Elimination to Belief Propagation

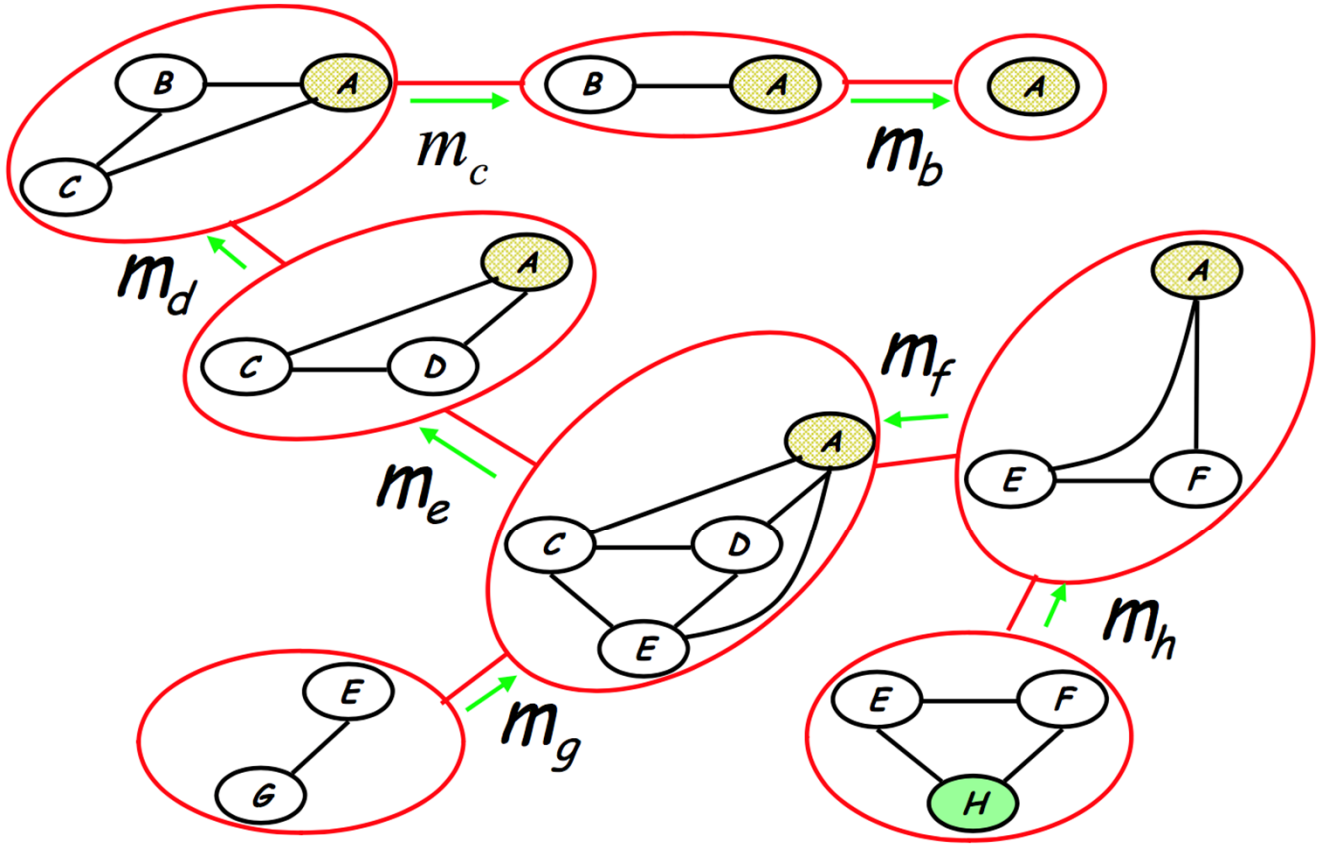


- Recall that induced dependency during marginalization is captured in elimination cliques
  - Summation <-> elimination
  - Intermediate term <-> elimination clique

$$
\begin{aligned}
& P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)\phi_g(e)\phi_h(e,f) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)P(e|c,d)\phi_f(a,e) \\
\Rightarrow\ & P(a)P(b)P(c|b)P(d|a)\phi_e(a,c,d) \\
\Rightarrow\ & P(a)P(b)P(c|b)\phi_d(a,c) \\
\Rightarrow\ & P(a)P(b)\phi_c(a,b) \\
\Rightarrow\ & P(a)\phi_b(a) \\
\Rightarrow\ & \phi(a)
\end{aligned}
$$



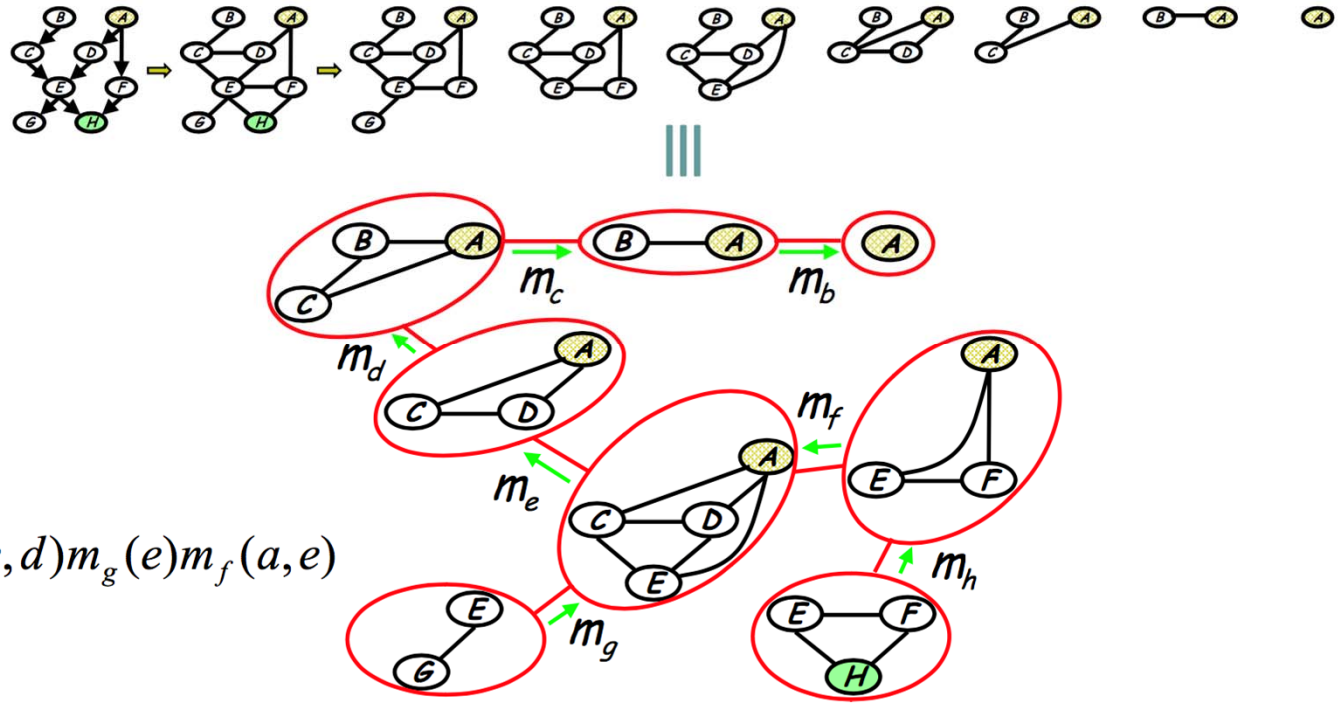
# A Clique Tree




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

# From Elimination to Message Passing

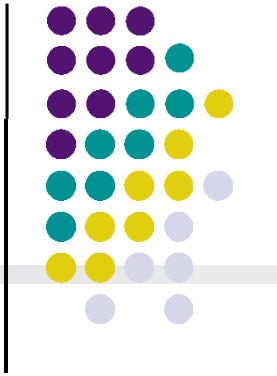
- Elimination = message passing on a clique tree




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

- Messages can be reused

# From Elimination to Message Passing

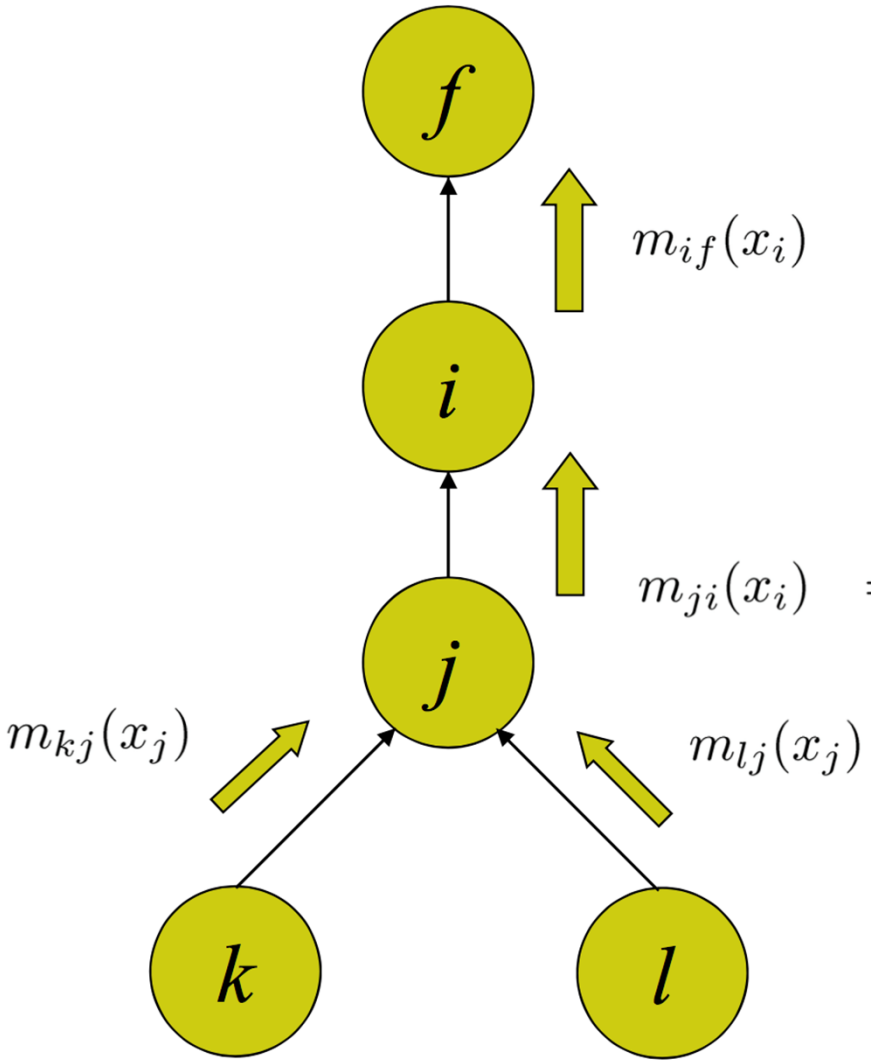


- Recall ELIMINATION algorithm:
  - Choose an ordering Z in which query node f is the final node
  - Place all potentials on an active list
  - Eliminate node i by removing all potential containing i, take sum/product over $x_i$
  - Place the resultant factor back on the list

- For a TREE graph:
  - Choose query node f as the root of the tree
  - View tree as a directed tree with edges pointing towards from f
  - Elimination ordering based on depth-first traversal
  - Elimination of each node can be considered as message-passing (or Belief Propagation) directly along tree branches, rather than on some transformed graphs

-> thus, we can use the tree itself as a data-structure to do general inference !!

# Message Passing for Trees




Let $m_{ij}(x_i)$ denote the factor resulting from eliminating variables from below up to i, which is a function of $x_i$:

$$m_{ji}(x_i) = \sum_{x_j} \left( \psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \right)$$

This is reminiscent of a message sent from j to i.

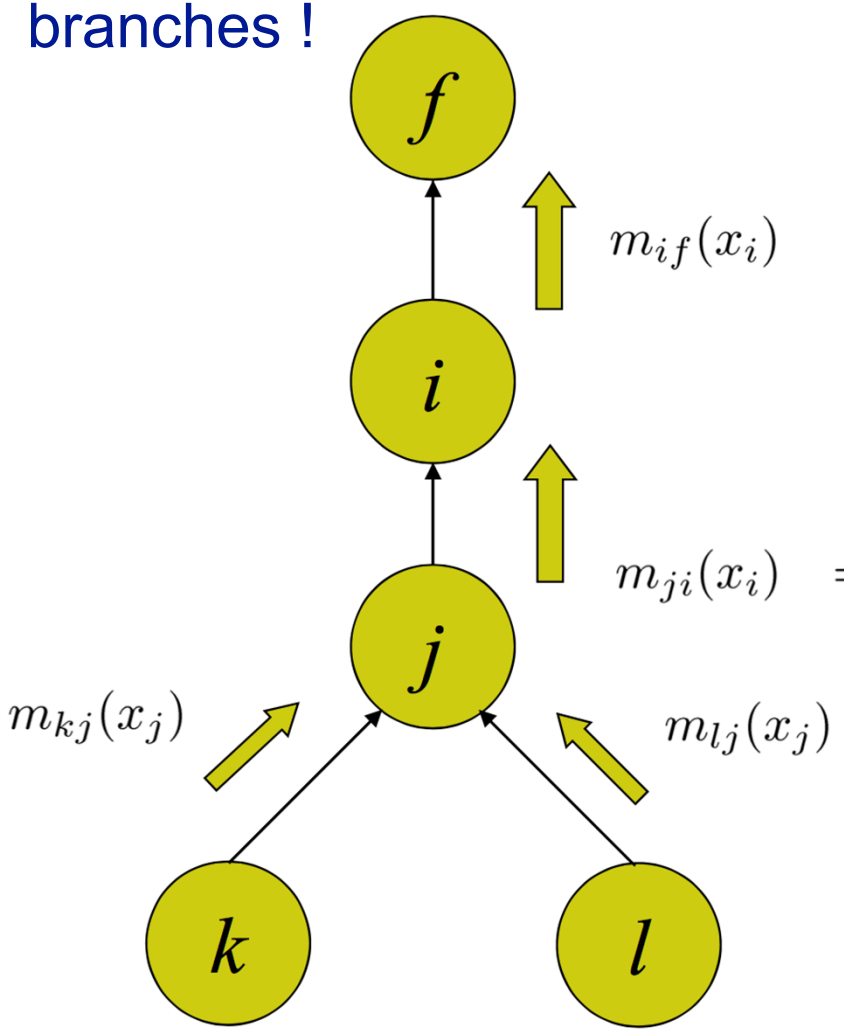
$$m_{ji}(x_i) = \sum_{x_j} \left( \psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \right)$$

$$p(x_f) \propto \psi(x_f) \prod_{e \in N(f)} m_e f(x_f)$$

$m_{ij}(x_i)$ represents a "belief" of $x_i$ from $x_j$ !

# Message Passing for Trees

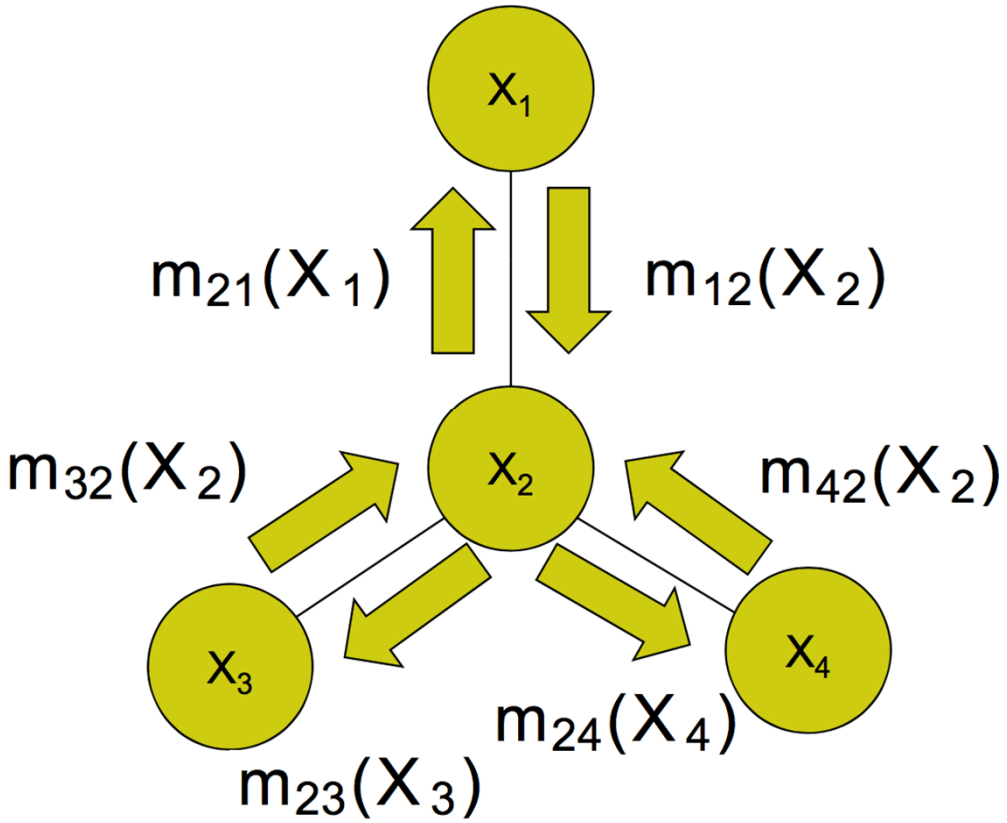
- Elimination on trees is equivalent to message passing along tree branches !



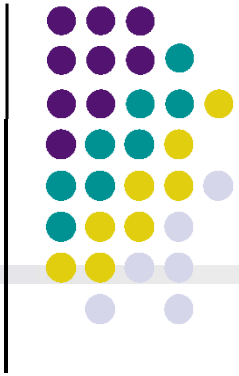

$$m_{ji}(x_i) = \sum_{x_j} \left( \psi(x_j)\psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_j) \right)$$

# The Message Passing Protocol

- A two-pass algorithm:



# Belief Propagation (SP-algorithm): Sequential Implementation



SUM-PRODUCT($\mathcal{T}$, $E$)
  EVIDENCE($E$)
  $f$ = CHOOSEROOT($\mathcal{V}$)
  **for** $e \in \mathcal{N}(f)$
    COLLECT($f, e$)
  **for** $e \in \mathcal{N}(f)$
    DISTRIBUTE($f, e$)
  **for** $i \in \mathcal{V}$
    COMPUTEMARGINAL($i$)

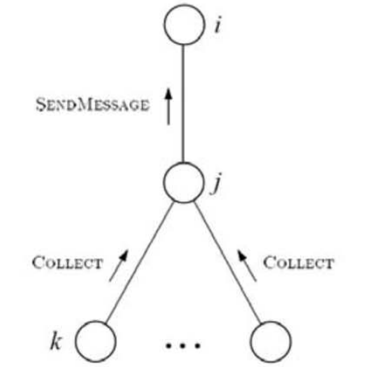
EVIDENCE($E$)
  **for** $i \in E$
    $\psi^E(x_i) = \psi(x_i)\delta(x_i, \bar{x}_i)$
  **for** $i \notin E$
    $\psi^E(x_i) = \psi(x_i)$

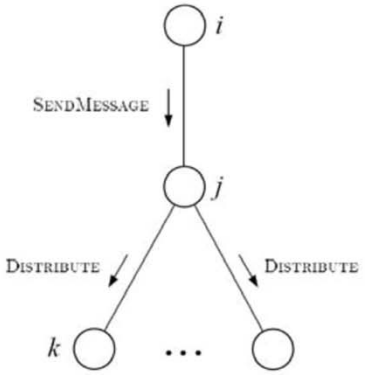
COLLECT($i, j$)
  **for** $k \in \mathcal{N}(j) \backslash i$
    COLLECT($j, k$)
  SENDMESSAGE($j, i$)

DISTRIBUTE($i, j$)
  SENDMESSAGE($i, j$)
  **for** $k \in \mathcal{N}(j) \backslash i$
    DISTRIBUTE($j, k$)

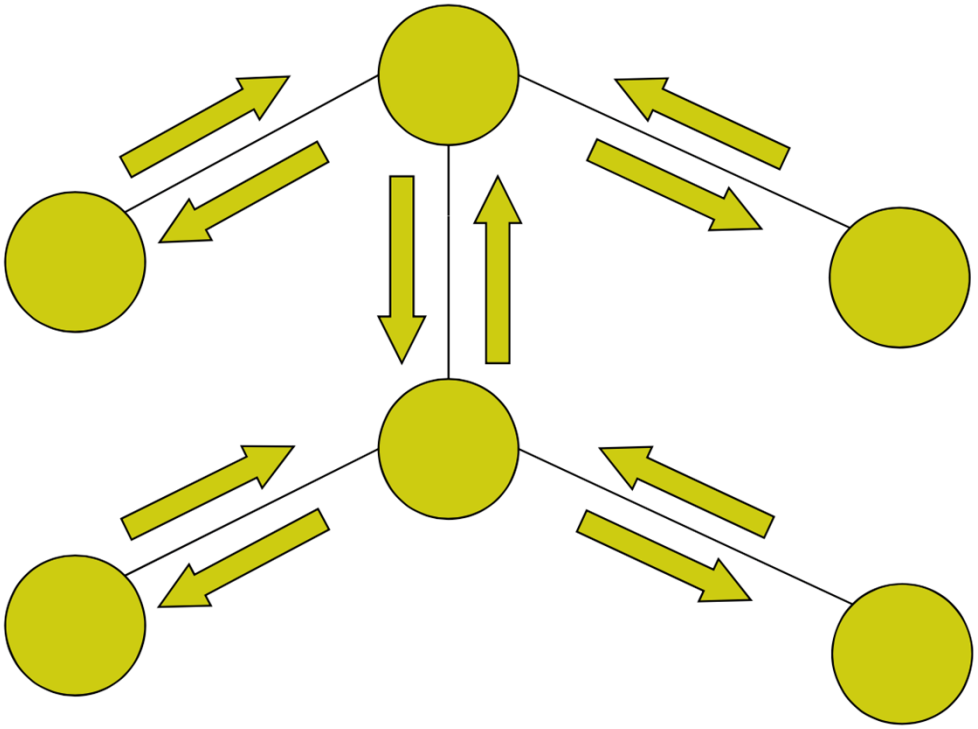
SENDMESSAGE($j, i$)
  $m_{ji}(x_i) = \sum_{x_j} (\psi^E(x_j)\psi(x_i, x_j) \prod_{k \in \mathcal{N}(j) \backslash i} m_{kj}(x_j))$

COMPUTEMARGINAL($i$)
  $p(x_i) \propto \psi^E(x_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)$



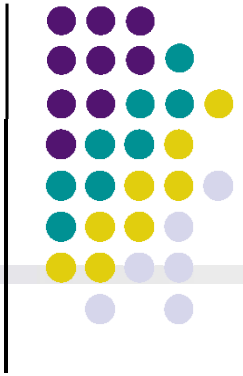



# Belief Propagation (SP-algorithm): Parallel Synchronous Implementation



- For a node of degree d, whenever messages have arrived on any subset of d-1 node, compute the message for the remaining edge and send !
  - A pair of messages have been computed for each edge, one for each direction
  - All incoming messages are eventually computed for each node

# Inference on General GM



- Now, what if the GM is not a tree-like graph ?

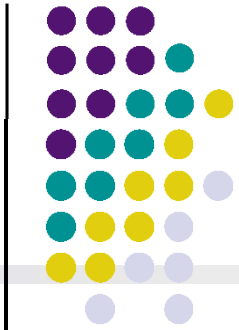
- Can we still directly run message message-passing protocol along its edges ?

- For non-trees, we do not have the guarantee that message-passing will be consistent !

- Then what ?
  - Construct a graph data-structure from P that has a tree structure, and run message-passing on it !

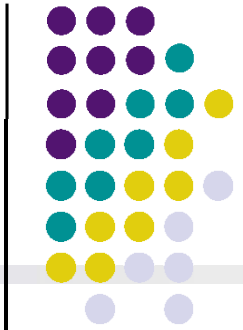
-> Junction tree algorithm

# A Sketch of the Junction Tree Algorithm



- The algorithm
  - Construction of junction trees — a special **clique tree**
  - Propagation of probabilities — a message-passing protocol
- Results in marginal probabilities of all cliques — solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: exponential in the size of the maximal clique — a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT
- Many well-known algorithms are special cases of JT

# Summary



- The simple Eliminate algorithm captures the key algorithmic operation underlying probabilistic inference:
  - That of taking a sum over product of potential functions
- 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 algorithms
- 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.