Bayesian Networks

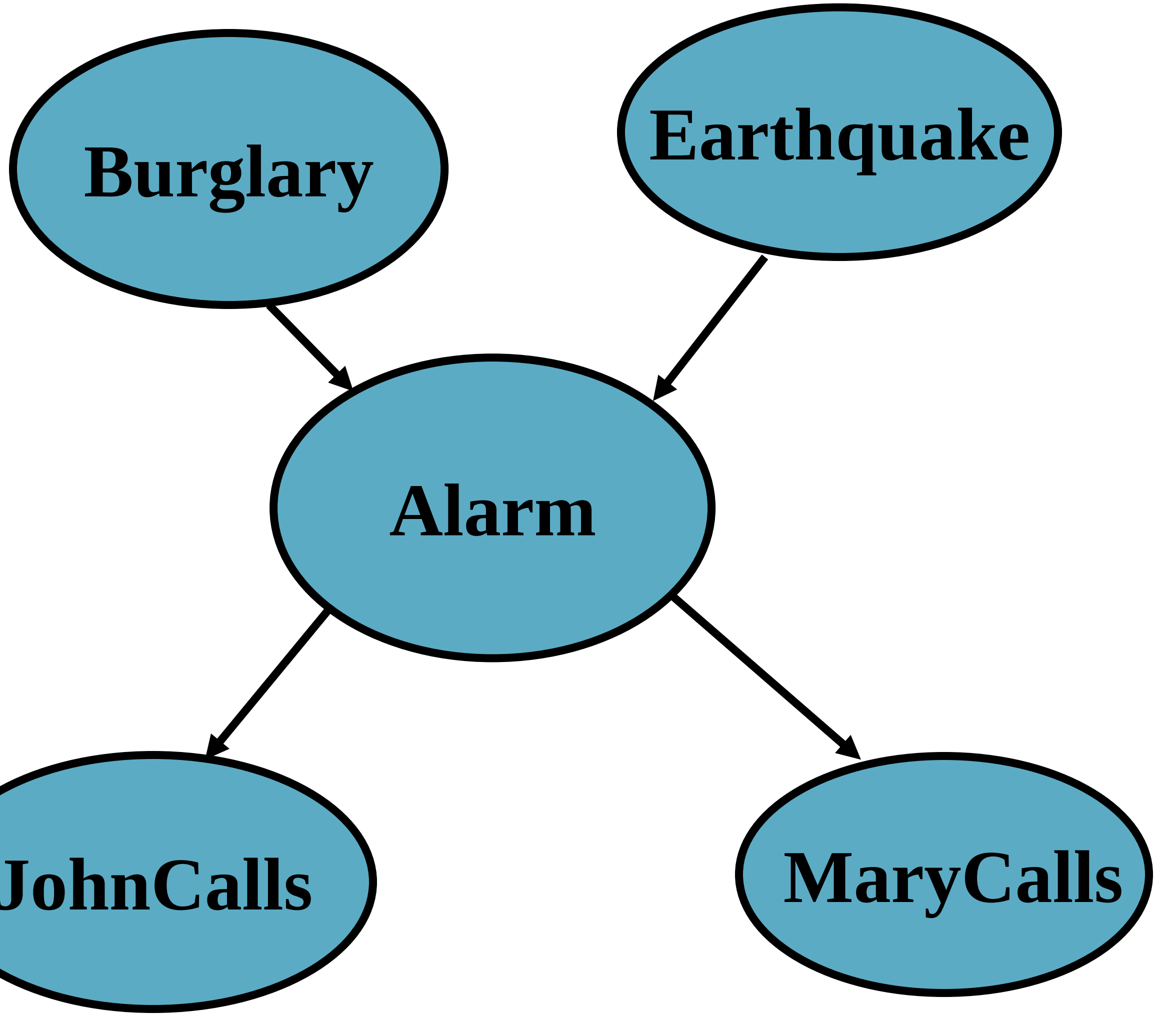
Dr. Jianlin Cheng

Department of Computer Science
University of Missouri, Columbia

Slides Adapted from Book and CMU, MU, Stanford Machine Learning Courses

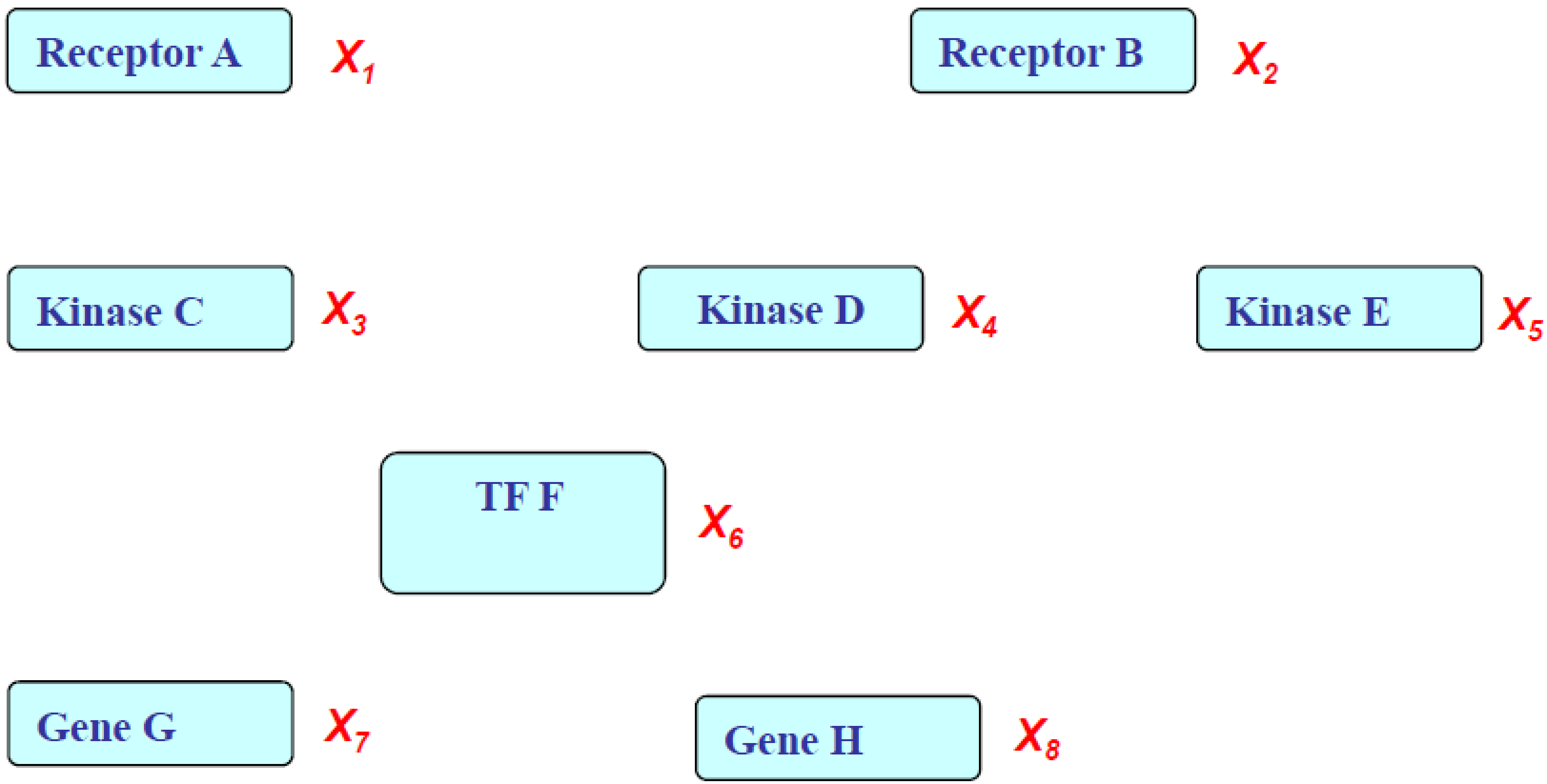
Fall, 2011

What is a Bayesian Network?



What is a Bayesian Network?

- A possible world for cellular signal transduction:

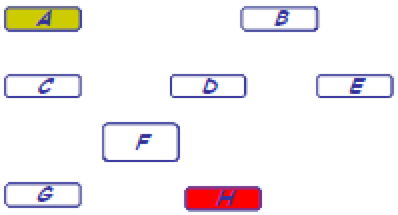


Basic Probability Concepts

- 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?

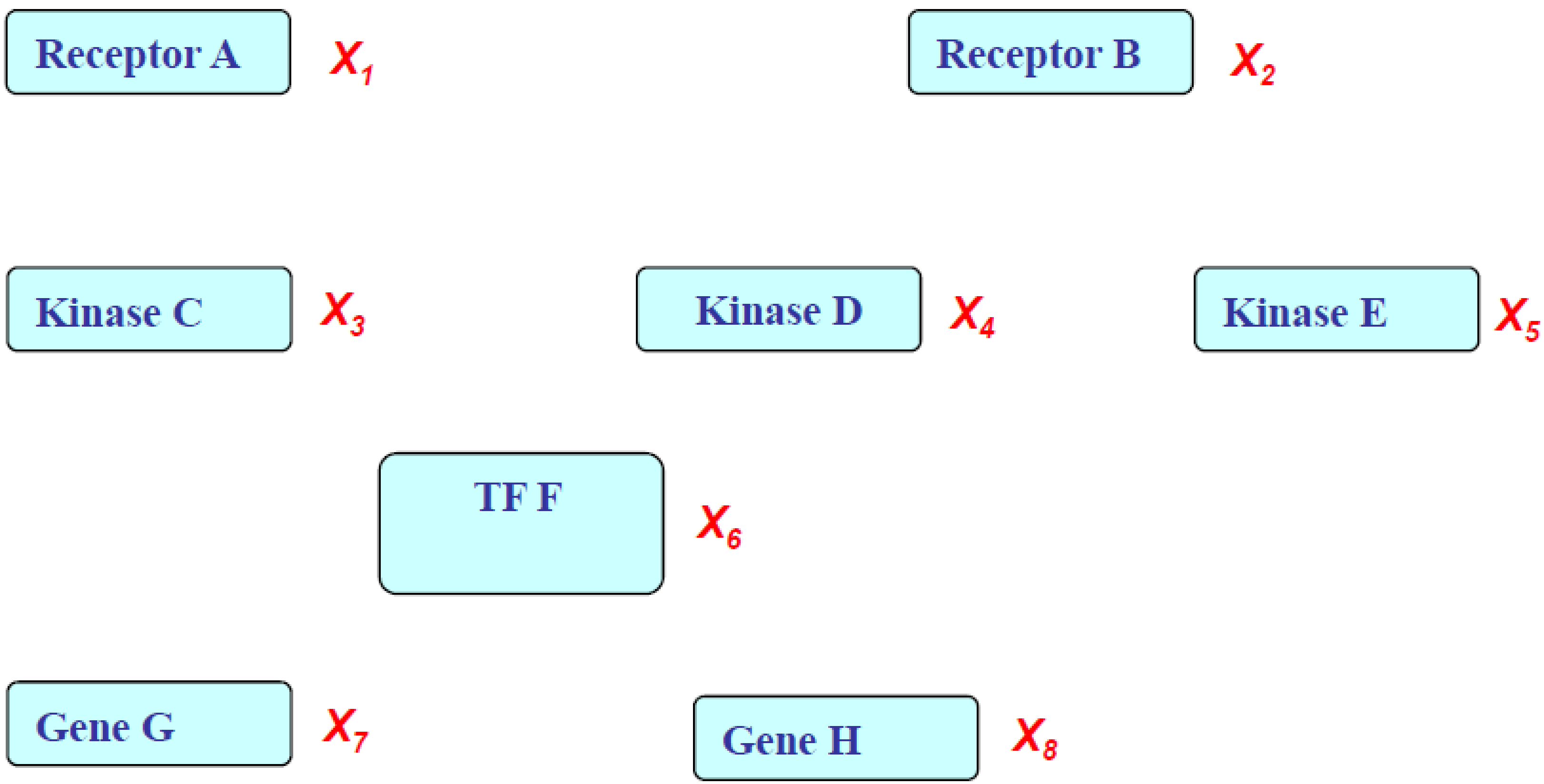


- Learning: where do we get all this probabilities?
  - Maximal-likelihood estimation? but how many data do we need?
  - Where do we put domain knowledge in terms of plausible relationships between variables, and plausible values of the probabilities?

- 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

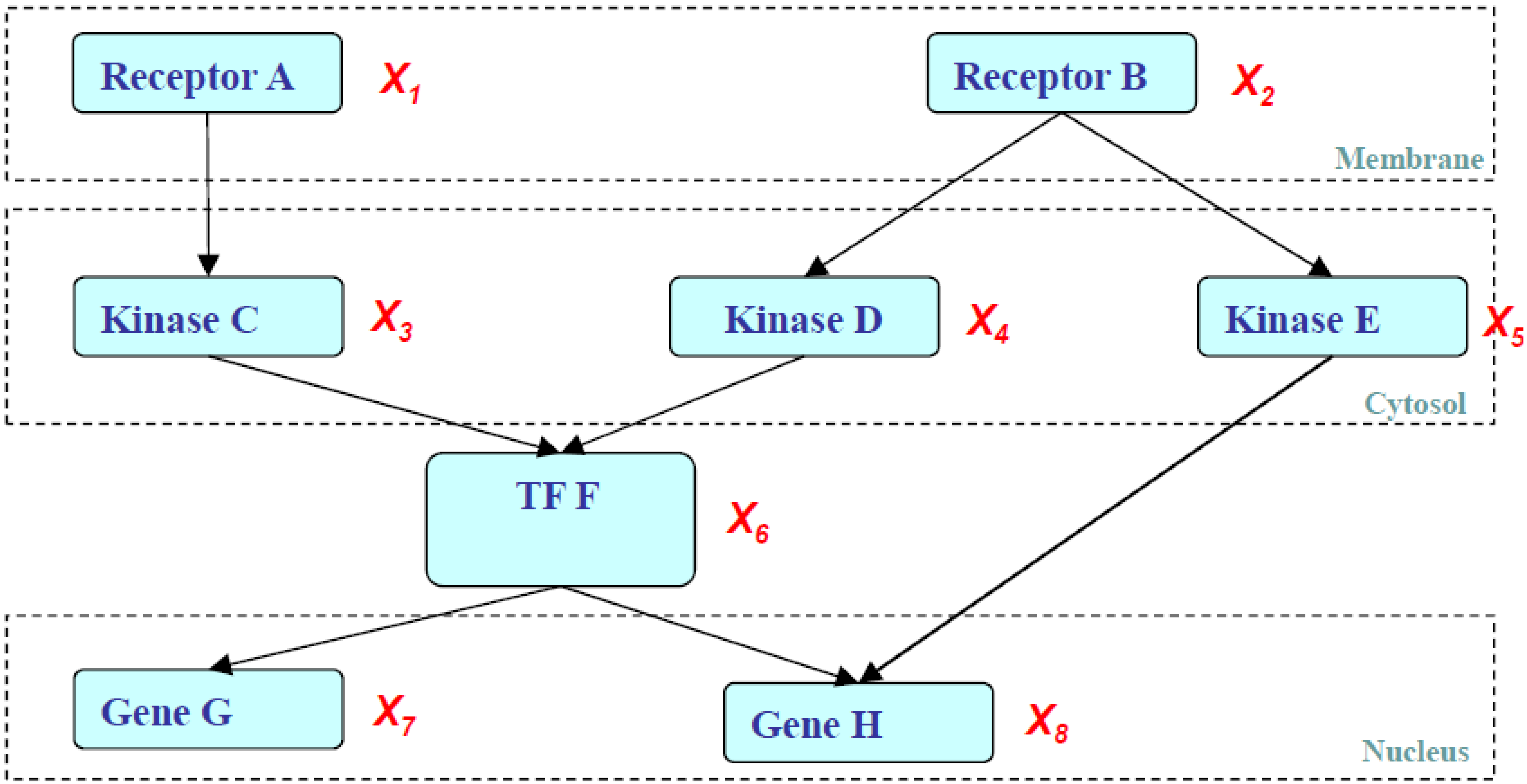
What is a Bayesian Network?

- A possible world for cellular signal transduction:



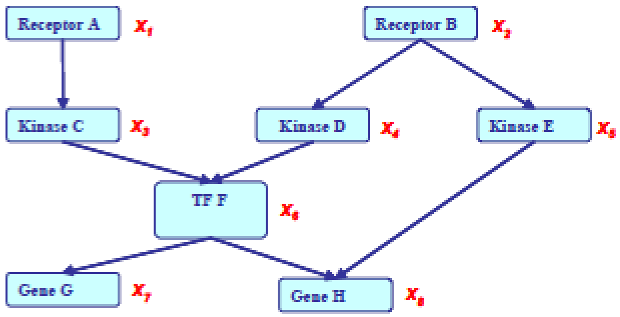
BN: Structure Simplify Representations

- Dependencies among variables



Bayesian Networks

- 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.,

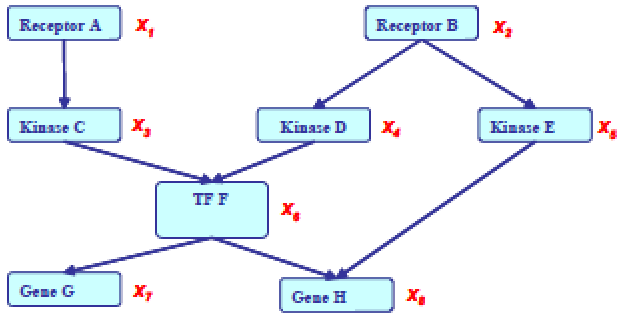


$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8)$$
$$= P(X_1)\, P(X_2)\, P(X_3 | X_1)\, P(X_4 | X_2)\, P(X_5 | X_2)\, P(X_6 | X_3, X_4)\, P(X_7 | X_6)\, P(X_8 | X_5, X_6)$$

- 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: 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":

$$P(\mathbf{X}) = \prod_i P(X_i \mid \mathbf{X}_{\pi_i})$$

  where $\mathbf{X}_{\pi_i}$ is the set of parents of xi. d is the number of nodes (variables) in the graph.

Proof

- $P(X_1, X_2, \ldots, X_d) = P(X_1|X_2, X_3, \ldots, X_d) * P(X_2, X_3, \ldots, X_d) = P(X_1|parent(X_1)) * P(X_2|X_3, \ldots, X_d) * P(X3, \ldots, Xd) = \ldots.$

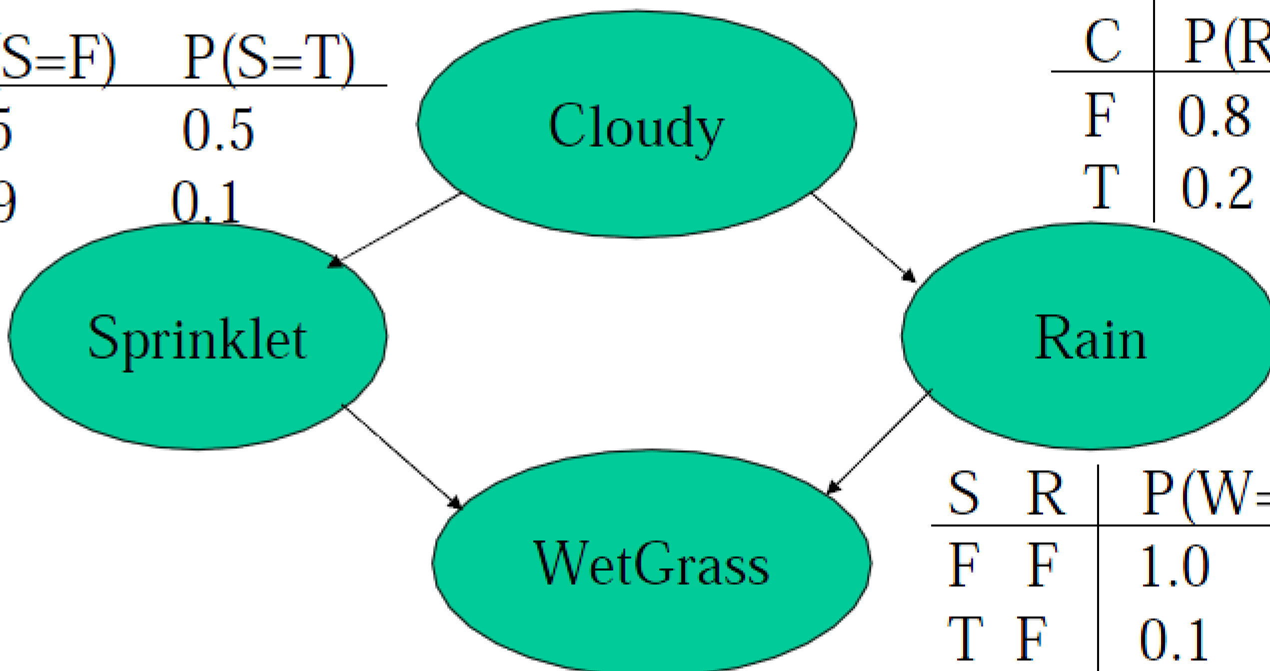
Conditional Probability Distribution

- Discrete variable: CPT, conditional probability table

P(C=F)	P(C=T)
0.5	0.5

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

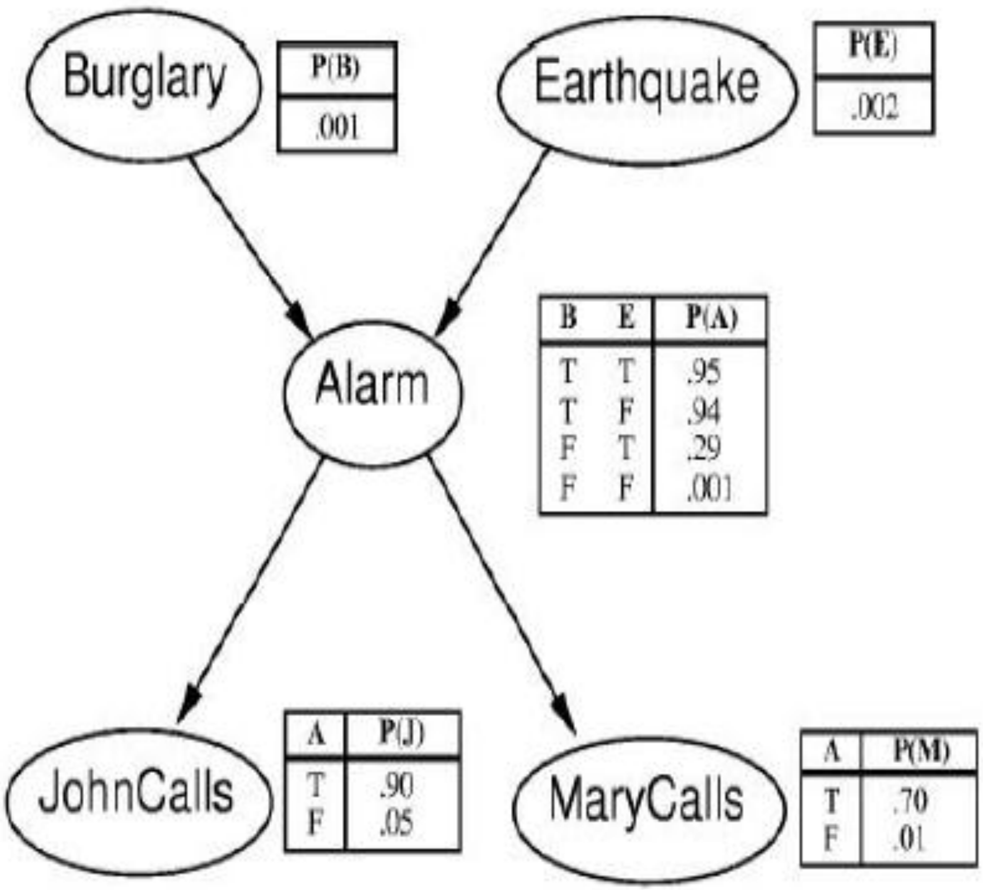
C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8



S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

P(C, S, R, W) = P(C) * P(S|C) * P(R|C,S) * P(W|C, S, R) = P(C) * P(S|C) * P(R|C) * P(W|S,R).

Examples


Burglary
P(B)
.001
Earthquake
P(E)
.002
Alarm
B E P(A)
T T .95
T F .94
F T .29
F F .001
JohnCalls
A P(J)
T .90
F .05
MaryCalls
A P(M)
T .70
F .01

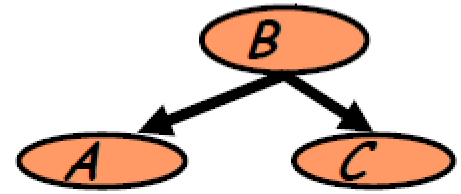
Qualitative Specification

- Where does the qualitative specification come from?
  - Prior knowledge of causal relationships
  - Prior knowledge of modular relationships
  - Assessment from experts
  - Learning from data
  - We simply link a certain architecture (e.g. a layered graph)
  - …

Local Structures and Independencies

- Common parent
  - Fixing B decouples A and C

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



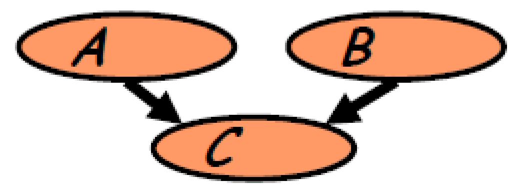
- Cascade
  - Knowing B decouples A and C

    "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

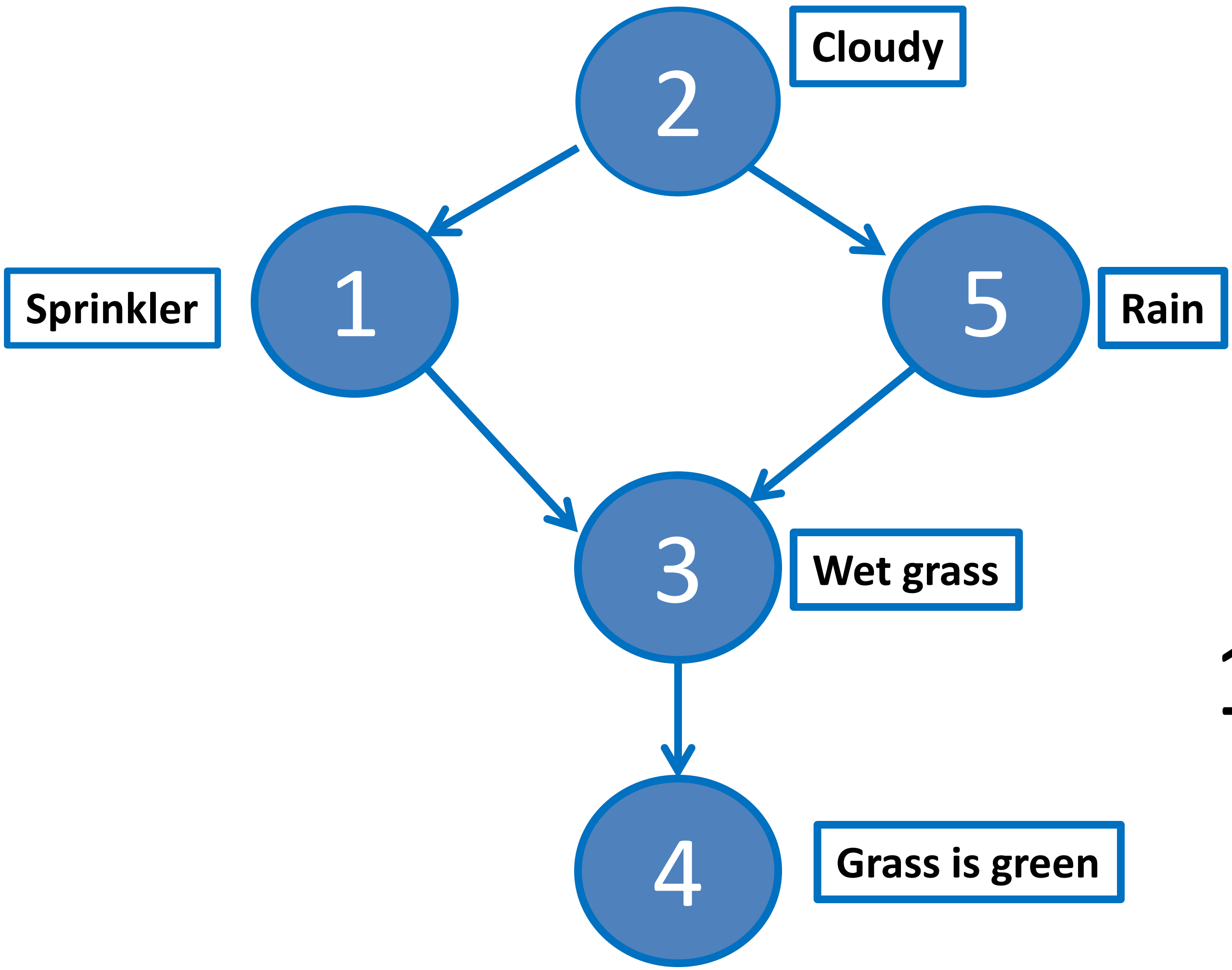
    because A can "explain away" B w.r.t. C

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



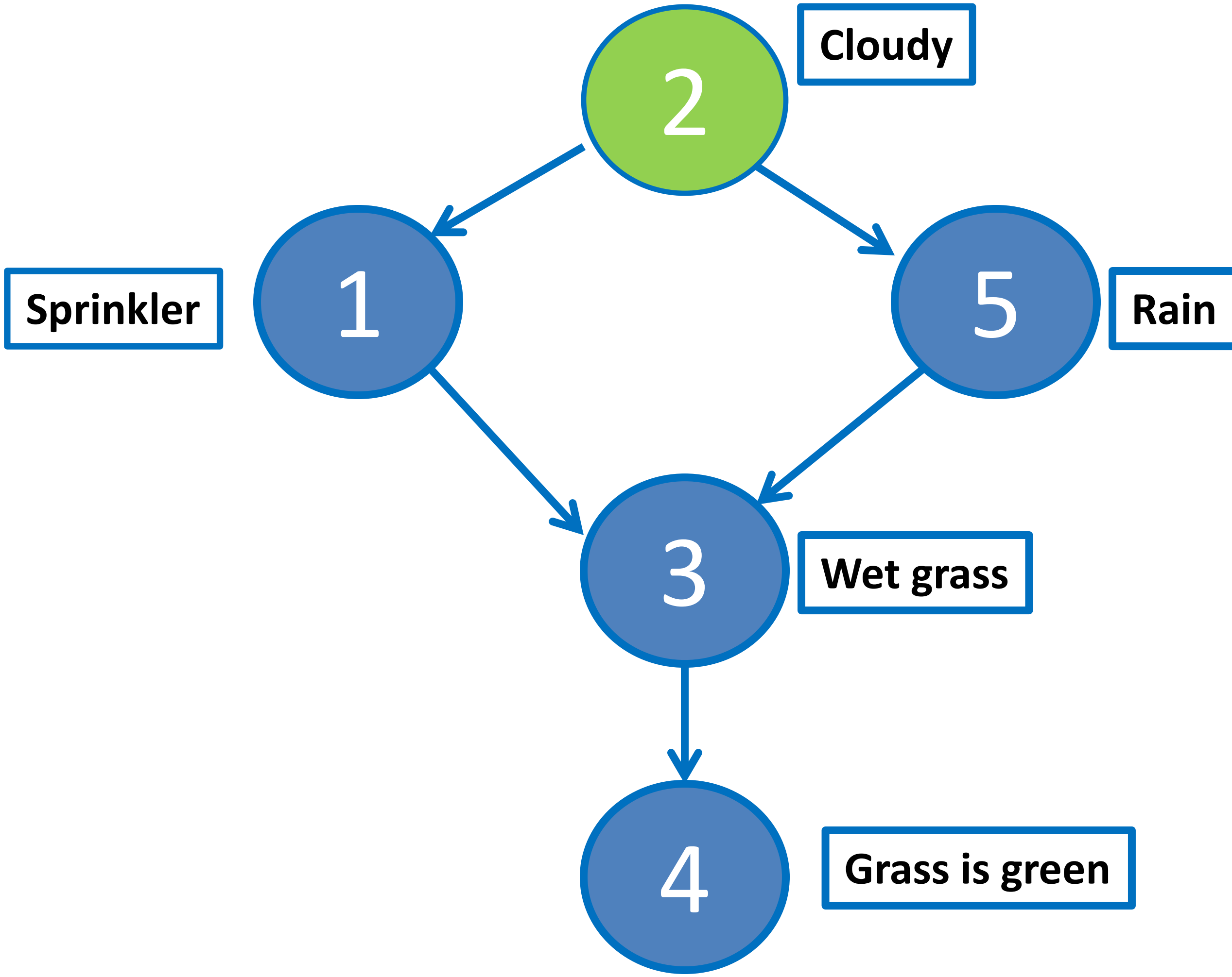
- The language is compact, the concepts are rich!

Assess Conditional Independence of Two Nodes in Bayesian Networks



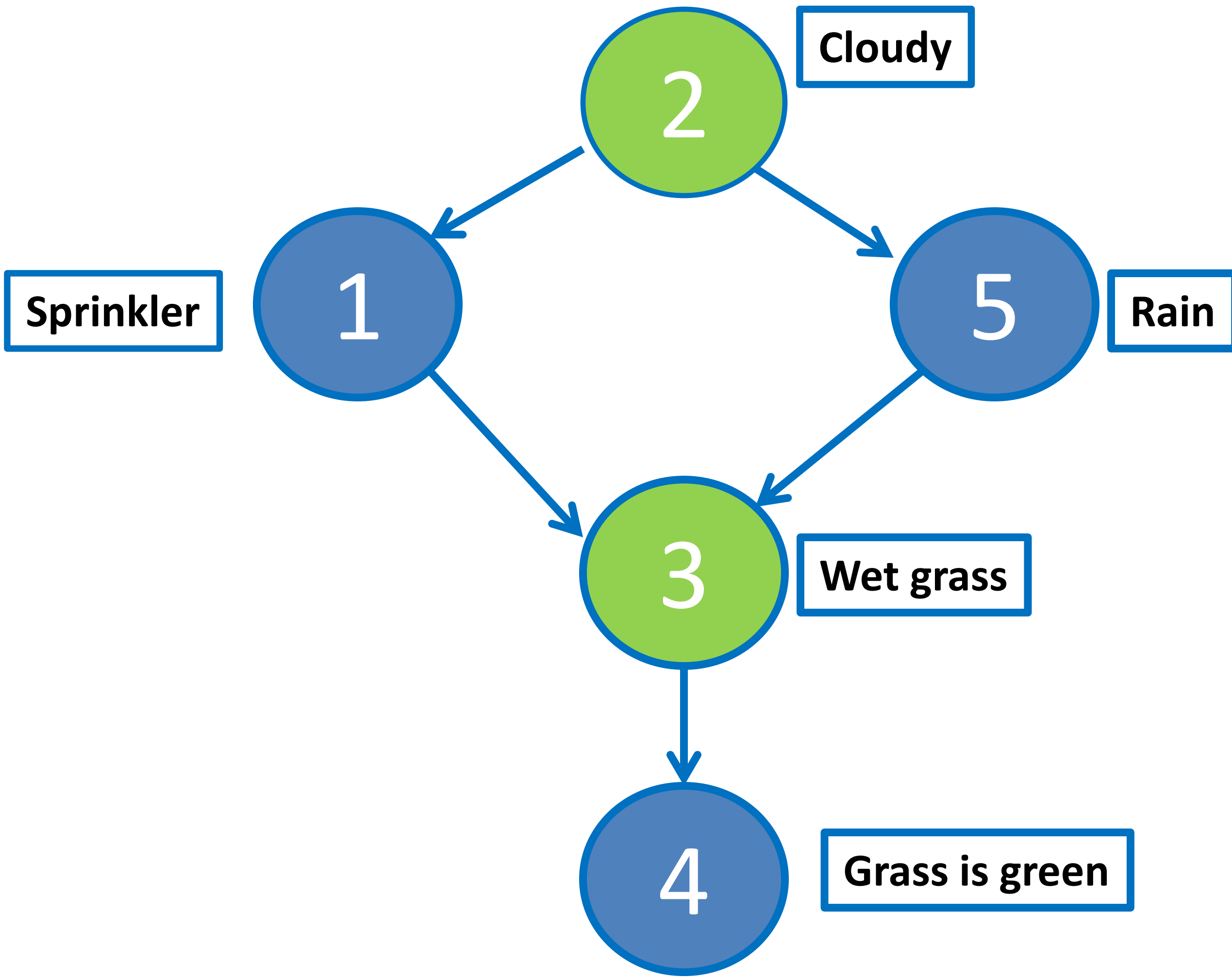
$1 \perp 5 \mid \phi$?

Assess Conditional Independence of Two Nodes in Bayesian Networks



$1 \perp 5 \mid 2$?

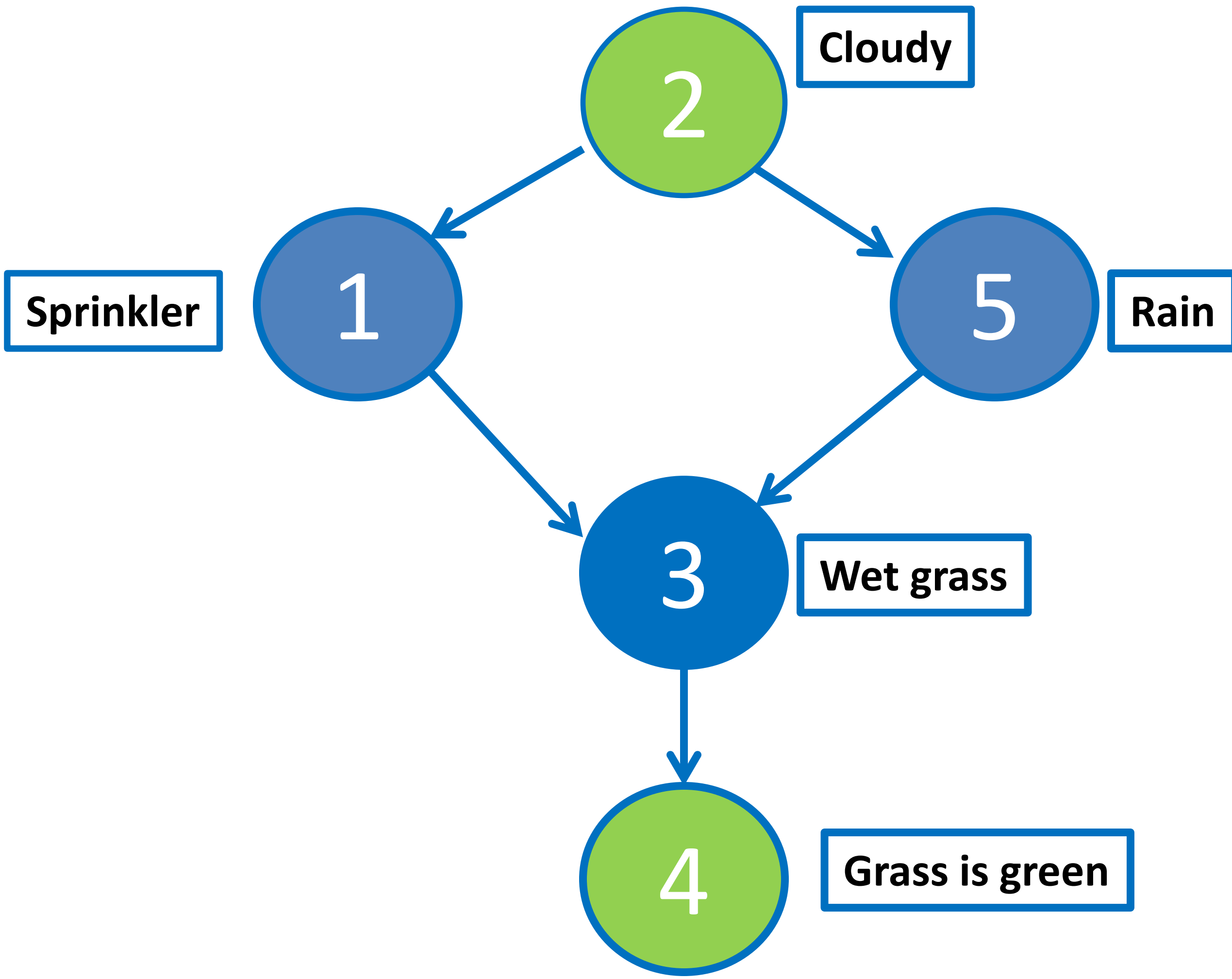
Assess Conditional Independence of Two Nodes in Bayesian Networks



$$1 \perp 5 \mid 2,3$$

?

Assess Conditional Independence of Two Nodes in Bayesian Networks



$$1 \perp 5 \mid 2,4$$

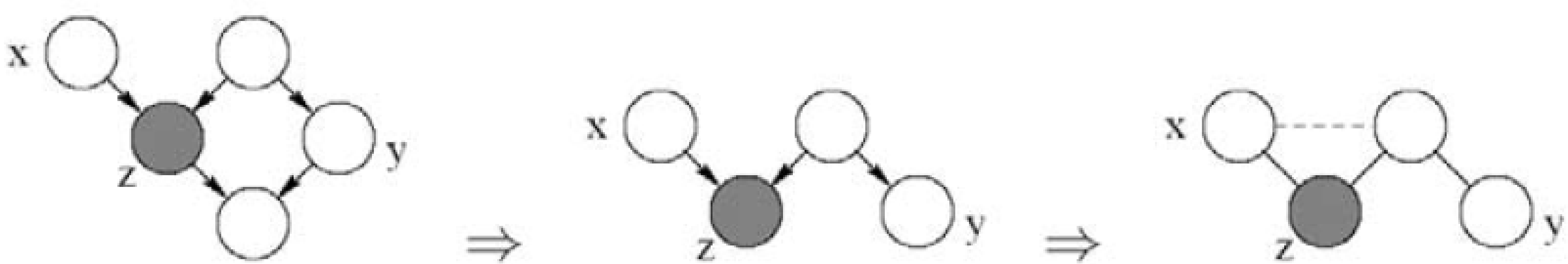
?

Graph Separation Criterion

- D-separation criterion for Bayesian networks (D for Directed edges):

  Definition: variables x and y are *D-separated* (conditionally independent) given z if they are separated in the *moralized* ancestral graph

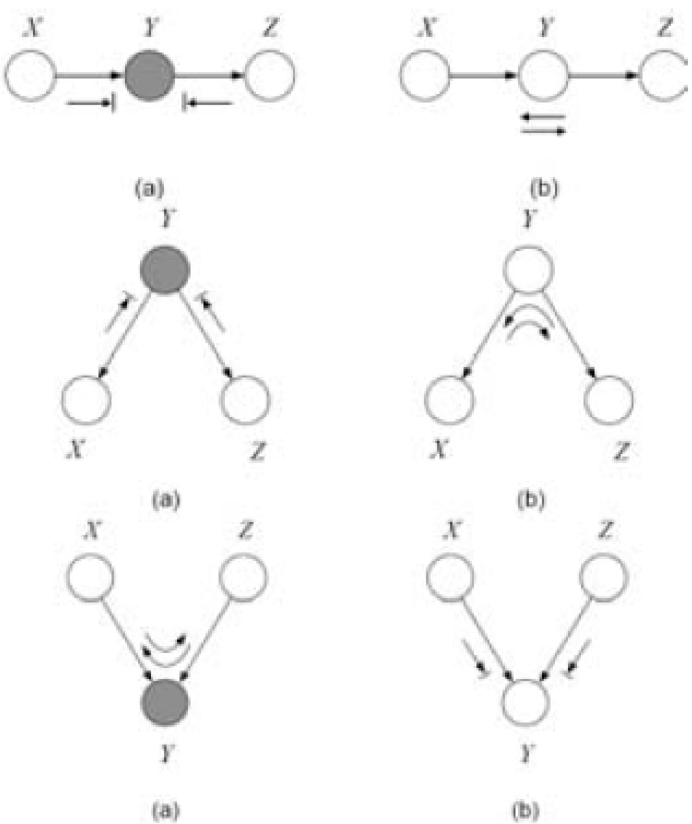
- Example:



original graph ancestral moral ancestral

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 bellow (and plus some boundary conditions):



- **Defn: $\mathcal{I}(G)$=all independence properties that correspond to d-separation:**

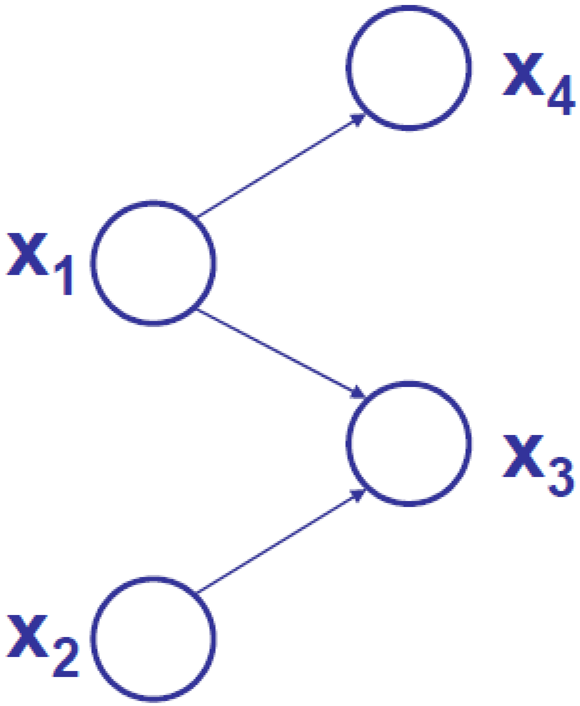
$$\mathrm{I}(G) = \left\{ X \perp Z \middle| Y : \mathrm{dsep}_G(X; Z|Y) \right\}$$

- **D-separation is sound and complete**

D-Separation Algorithm

- All the paths between two nodes must be D-Separated.
- **A -> B -> C** (linear, B is known, then the path is blocked)
- **A <- B -> C** (diverging, B is known, then the path is blocked)
- **A -> <u>B</u> <- C** (converging, B & and its descendants are **not** known)

An Example



- Complete the I(G) of this graph:

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

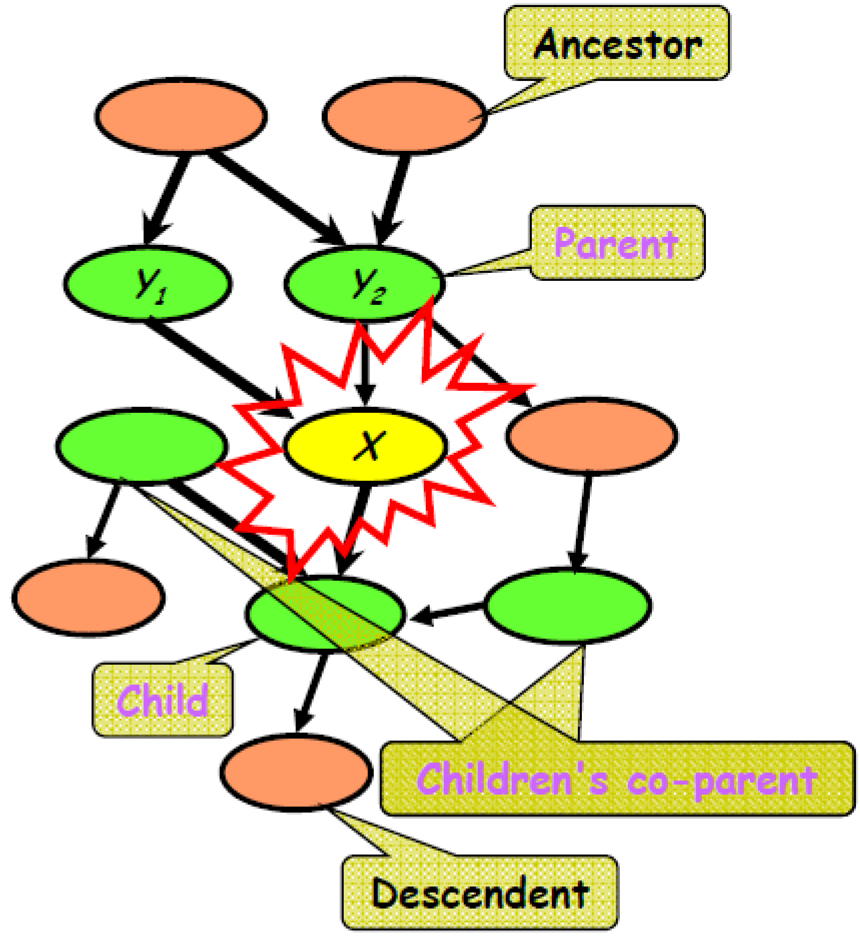
BN: Conditional Independence Semantics

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



Toward 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**

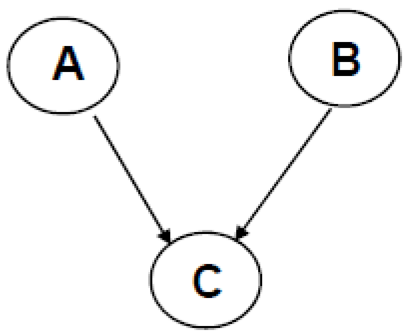
  For a graph G,

  Let $\mathcal{D}_1$ denote the family of all distributions that satisfy I(G),

  Let $\mathcal{D}_2$ denote the family of all distributions that factor according to G,

  Then $\mathcal{D}_1 \equiv \mathcal{D}_2$.

Quantitative Specification



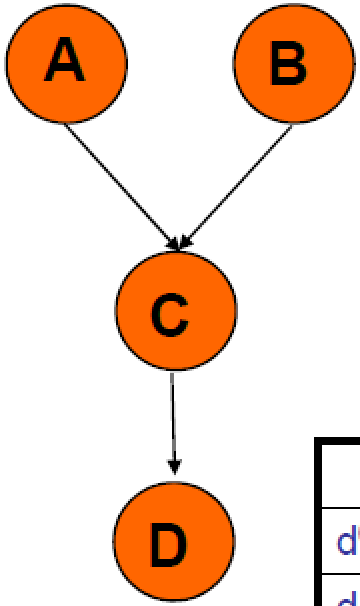
$p(A,B,C) =$

Conditional Probability Tables (CPTs)

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

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



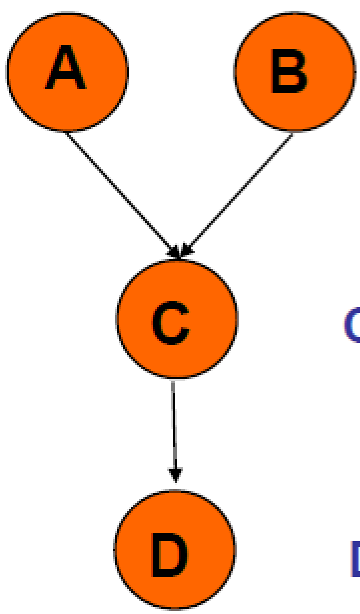
	a^0b^0	a^0b^1	a^1b^0	a^1b^1
c^0	0.45	1	0.9	0.7
c^1	0.55	0	0.1	0.3

	c^0	c^1
d^0	0.3	0.5
d^1	07	0.5

Conditional Probability Density Function (CPDs)

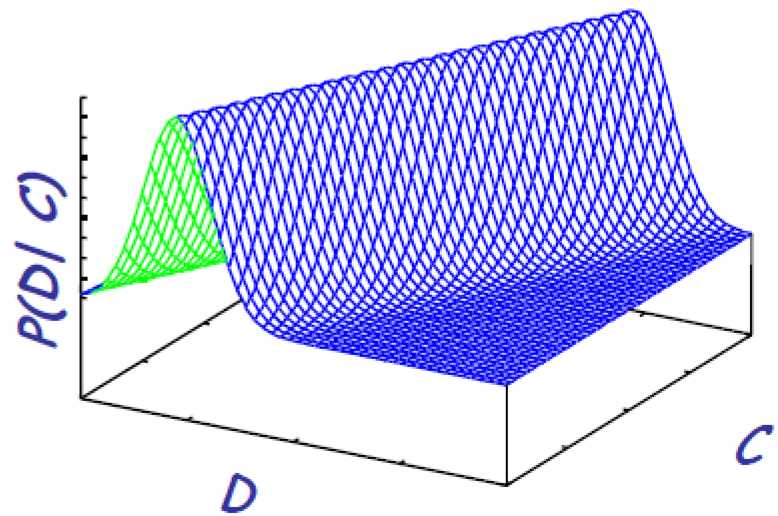
$A \sim N(\mu_a, \Sigma_a)$ $B \sim N(\mu_b, \Sigma_b)$

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

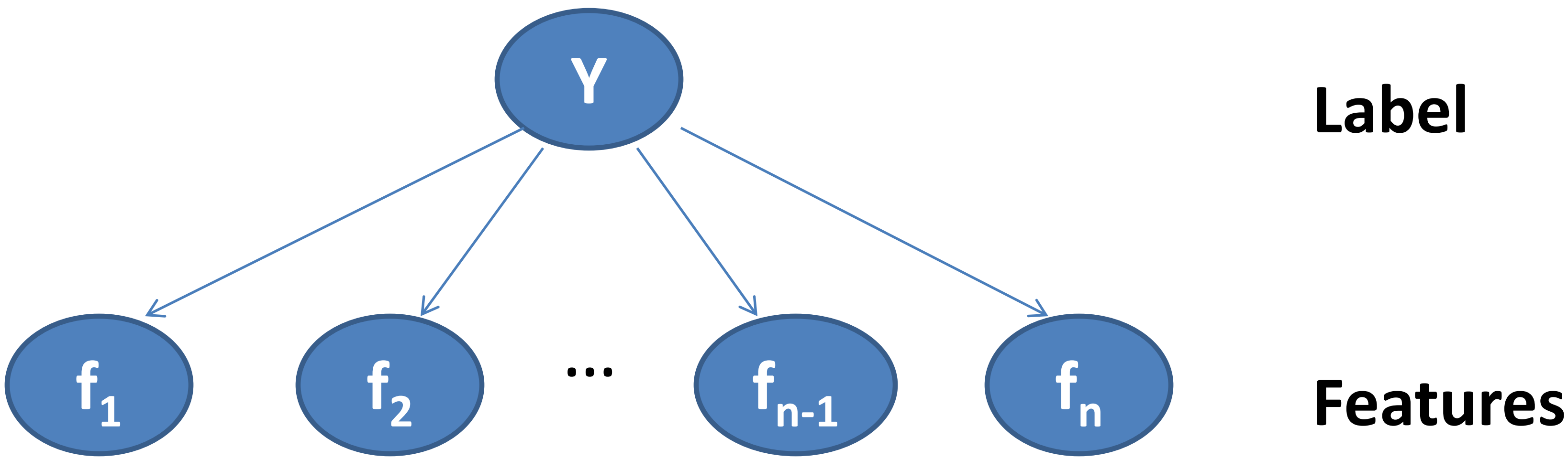


$C \sim N(A+B, \Sigma_c)$

$D \sim N(\mu_a + C, \Sigma_a)$



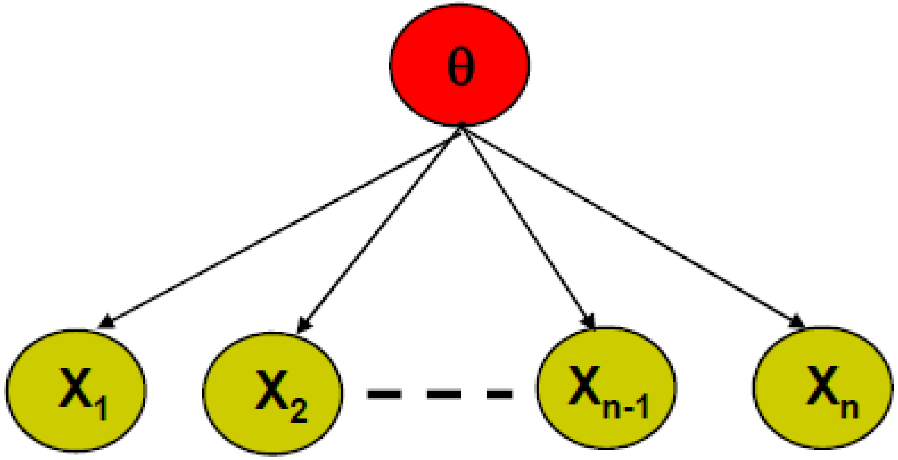
Conditional Independencies



What is the model?

a) **When Y is known?**
b) **When Y is not known?**

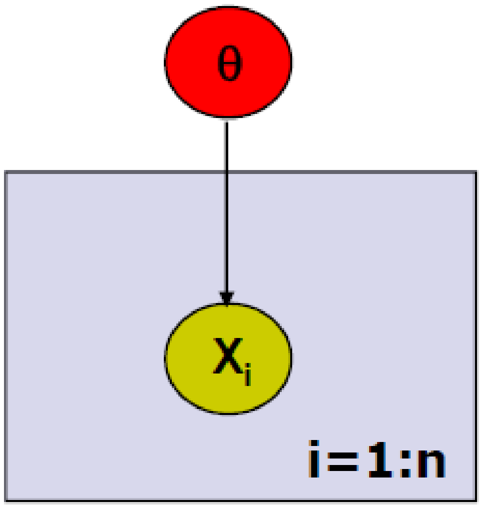
Conditional Independent Observations



Model parameters

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

“Plate” Notation



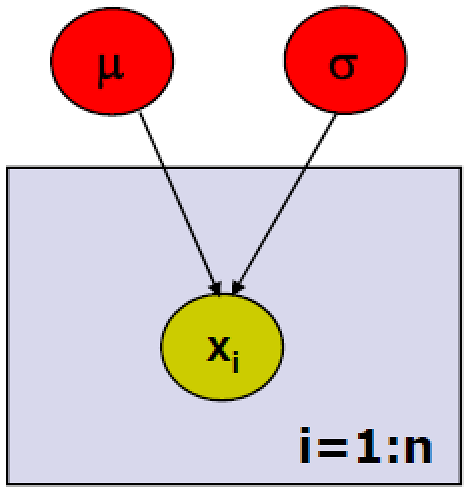
Model parameters

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:

$p(x_1,\ldots x_n \mid \mu, \sigma) \quad = P \;\; p(x_i \mid \mu, \sigma)$

$= \quad p(\text{data} \mid \text{parameters})$

$= \quad p(D \mid \theta)$

where $\theta = \{\mu, \sigma\}$

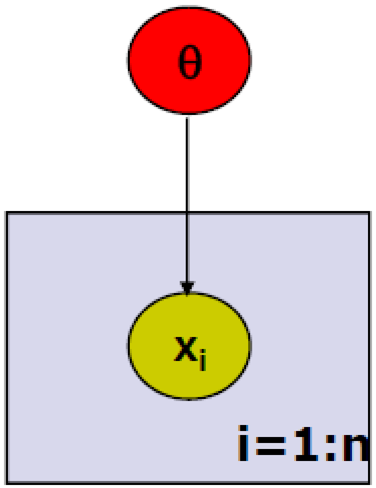
- Likelihood = p(data | parameters)

  $= p(\, D \mid \theta\,)$

  $= L\, (\theta)$

- Likelihood tells us how likely the observed data are conditioned on a particular setting of the parameters
  - Often easier to work with log $L\, (\theta)$

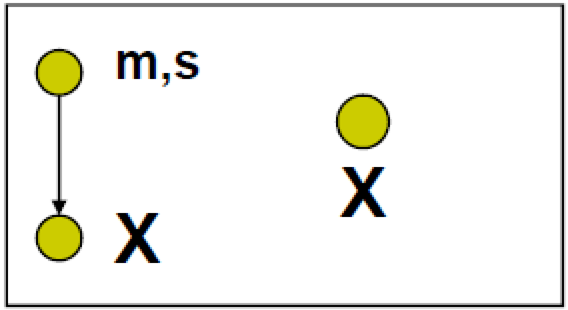
Bayesian Model



More Examples

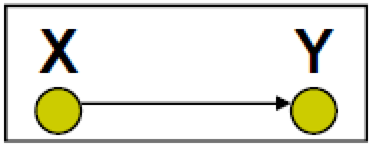
Density estimation

Parametric and nonparametric methods



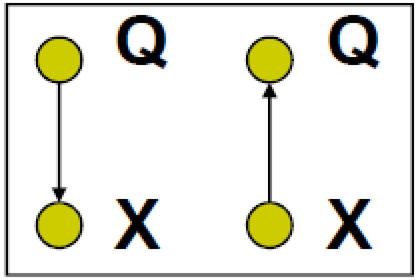
Regression

Linear, conditional mixture, nonparametric



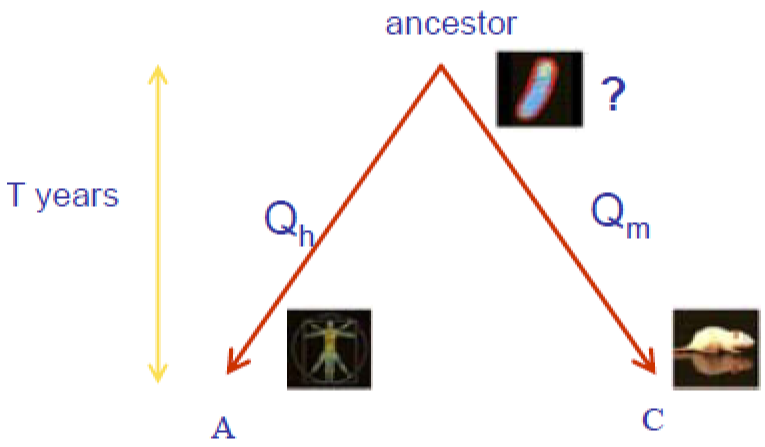
Classification

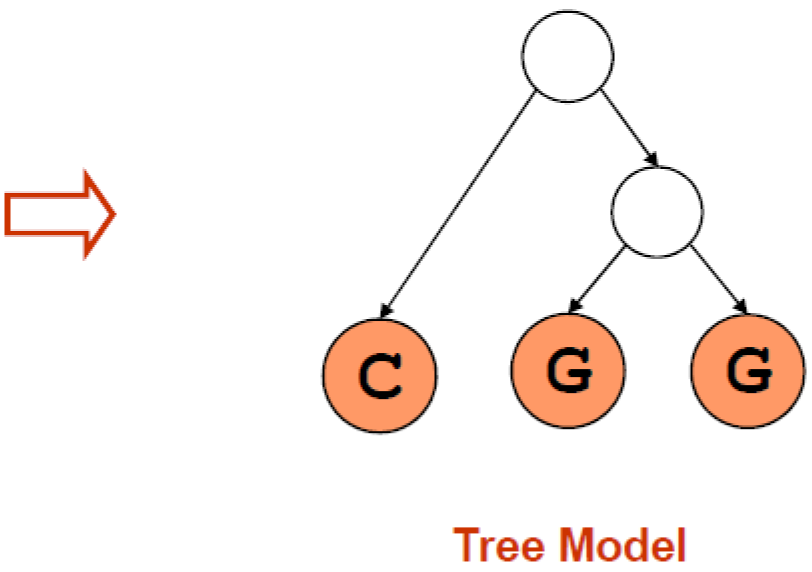
Generative and discriminative approach



Example, Con'd

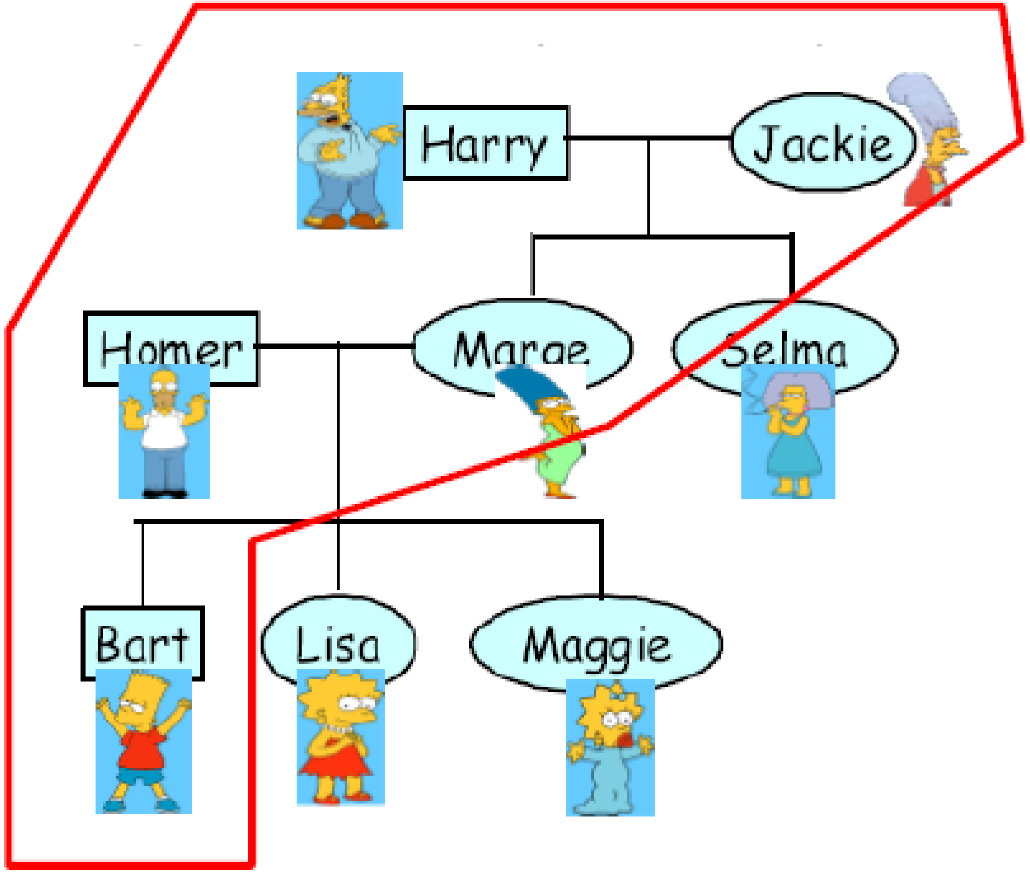
- Evolution





Example, Con'd

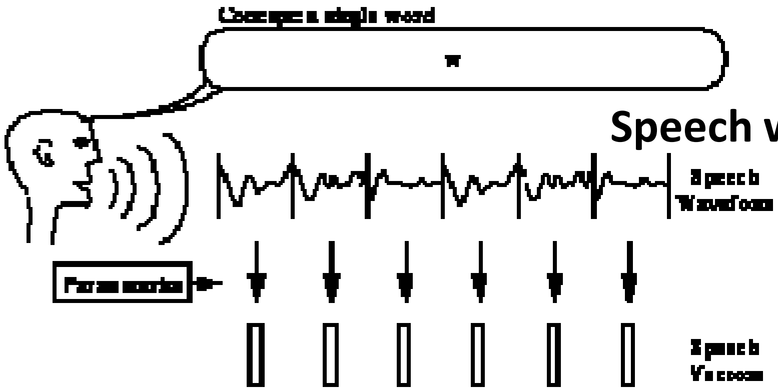
- Genetic Pedigree



Example, Con'd

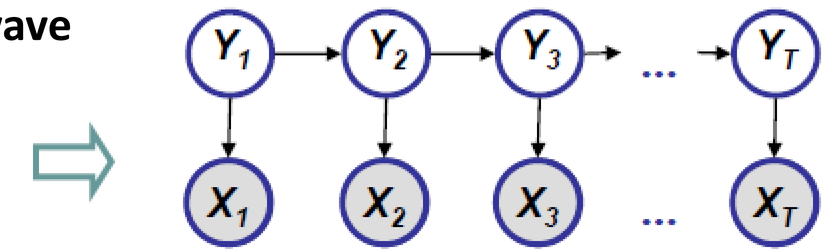
- Speech recognition

Word



Speech wave

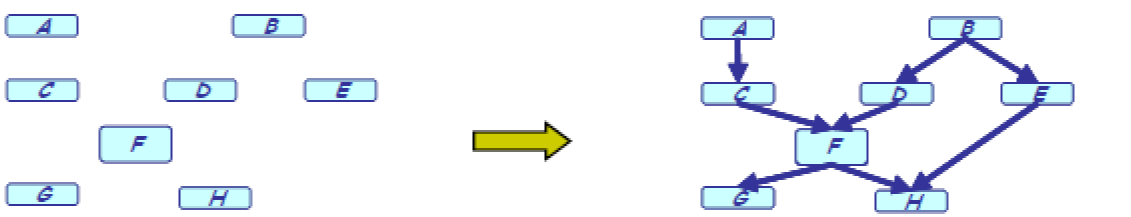
Phoneme (spectral code)



Hidden Markov Model

BN and Graphical Models

- A Bayesian network is a special case of **Graphical Models**
- A Graphical Model refers to a family of distributions on a set of random variables that are compatible with all the probabilistic independence propositions encoded by a graph that connects these variables
- It is a smart way to write/specify/compose/design exponentially-large probability distributions without paying an exponential cost, and at the same time endow the distributions with structured semantics



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

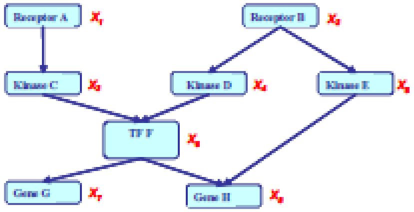
$$P(X_{18}) = P(X_1)P(X_2)P(X_3 \mid X_1,X_2)P(X_4 \mid X_2)P(X_5 \mid X_2)$$
$$P(X_6 \mid X_3,X_4)P(X_7 \mid X_6)P(X_8 \mid X_5,X_6)$$

Two Types of GMs

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

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

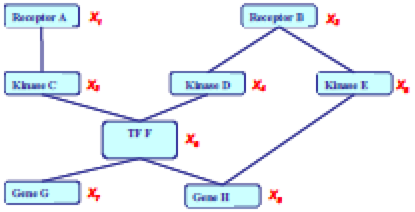
$$= P(X_1)\, P(X_2)\, P(X_3 | X_1)\, P(X_4 | X_2)\, P(X_5 | X_2)\, P(X_6 | X_3, X_4)\, P(X_7 | X_6)\, P(X_8 | X_5, X_6)$$



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

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

$$= 1/Z \exp\{E(X_1)+E(X_2)+E(X_3, X_1)+E(X_4, X_2)+E(X_5, X_2) + E(X_6, X_3, X_4)+E(X_7, X_6)+E(X_8, X_5, X_6)\}$$



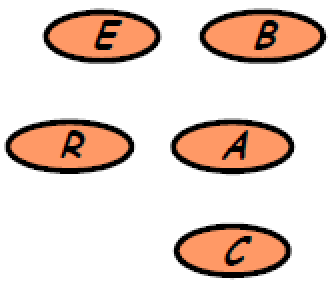
Probabilistic Inference

- Computing statistical queries regarding the network, e.g.:
  - Is node X independent on node Y given nodes Z,W ?
  - What is the probability of X=true if (Y=false and Z=true)?
  - What is the joint distribution of (X,Y) if Z=false?
  - What is the likelihood of some full assignment?
  - What is the most likely assignment of values to all or a subset the nodes of the network?
- General purpose algorithms exist to fully automate such computation
  - Computational cost depends on the topology of the network
  - Exact inference:
    - The junction tree algorithm
  - Approximate inference;
    - Loopy belief propagation, variational inference, Monte Carlo sampling

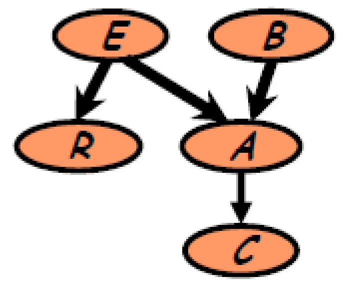
Learning in BN

The goal:

Given set of independent samples (*assignments* of random variables), find the *best* (the most likely?) Bayesian Network (both DAG and CPDs)



(B,E,A,C,R)=(T,F,F,T,F)
(B,E,A,C,R)=(T,F,T,T,F)
....
(B,E,A,C,R)=(F,T,T,T,F)

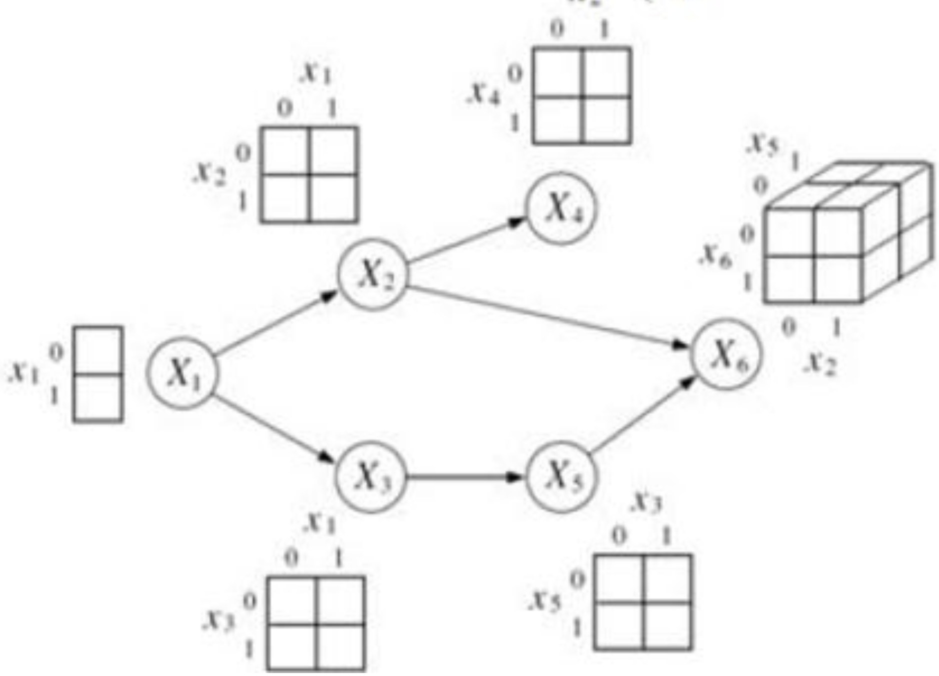


E	B	P(A | E,B)	
e	b	0.9	0.1
e	$\bar{b}$	0.2	0.8
$\bar{e}$	b	0.9	0.1
$\bar{e}$	$\bar{b}$	0.01	0.99

MLE Learning

- If we assume the parameters for each CPD are globally independent, and all nodes are **fully observed**, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\ell(\theta;D) = \log p(D \mid \theta) = \log \prod_{n} \left(\prod_{i} p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right) = \sum_{i} \left(\sum_{n} \log p(x_{n,i} \mid \mathbf{x}_{n,\pi_i}, \theta_i) \right)$$

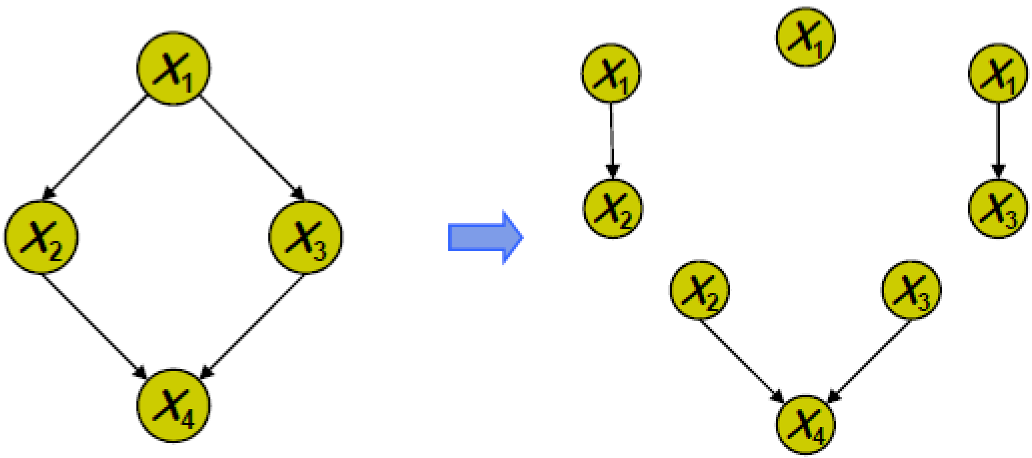


Example: Decomposable likelihood of a directed model

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1)\, p(x_2 \mid x_1, \theta_1)\, p(x_3 \mid x_1, \theta_3)\, p(x_4 \mid x_2, x_3, \theta_1)$$

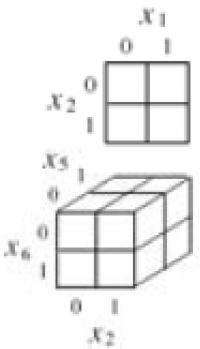
- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.



MLEs for BNs with Tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

$$\theta_{ijk} \stackrel{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k)$$



  - Note that in case of multiple parents, $\mathbf{X}_{\pi_i}$ will have a composite state, and the CPD will be a high-dimensional table
  - The sufficient statistics are counts of family configurations

$$n_{ijk} \stackrel{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k$$

- The log-likelihood is $\ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk}$

- Using a Lagrange multiplier to enforce $\sum_j \theta_{ijk} = 1$, we get:

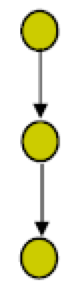
$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}}$$

An Example

- Three variables: C – Cloudy, R – Rain, S – Sprinkler
- Data: (C=T, R = T, S = F), (C = T, R = F, S = F), (C = F, R = F, S = T)
- P(C = T) = ?, P(C = F) = ?
- P(R = T | C = T) = ? P(R = F | C = F) = ?
- P(S = T | C = T) = ?, P(S = T | C = F) = ?

Summary

- Represent dependency structure with a directed acyclic graph
  - Node <-> random variable
  - Edges encode dependencies
    - Absence of edge -> conditional independence
  - Plate representation
  - A BN 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
- Support efficient inference and learning

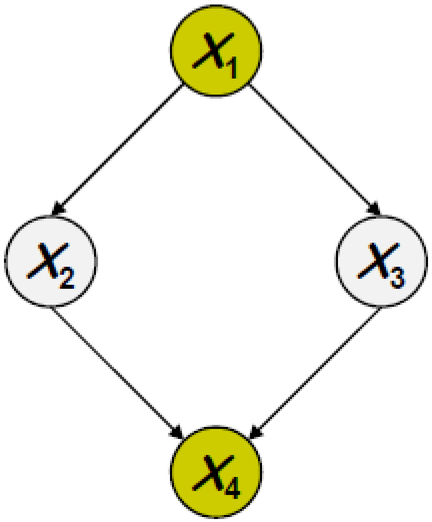
Inference and Learning

- We now have compact representations of probability distributions: **BN**
- 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*?
    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*.

What if some nodes are not observed?

- Consider the distribution defined by the directed acyclic GM:

$$p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_1)$$



- Need to compute $p(x_H|x_V)$ → inference

Inferential Query 1: Likelihood

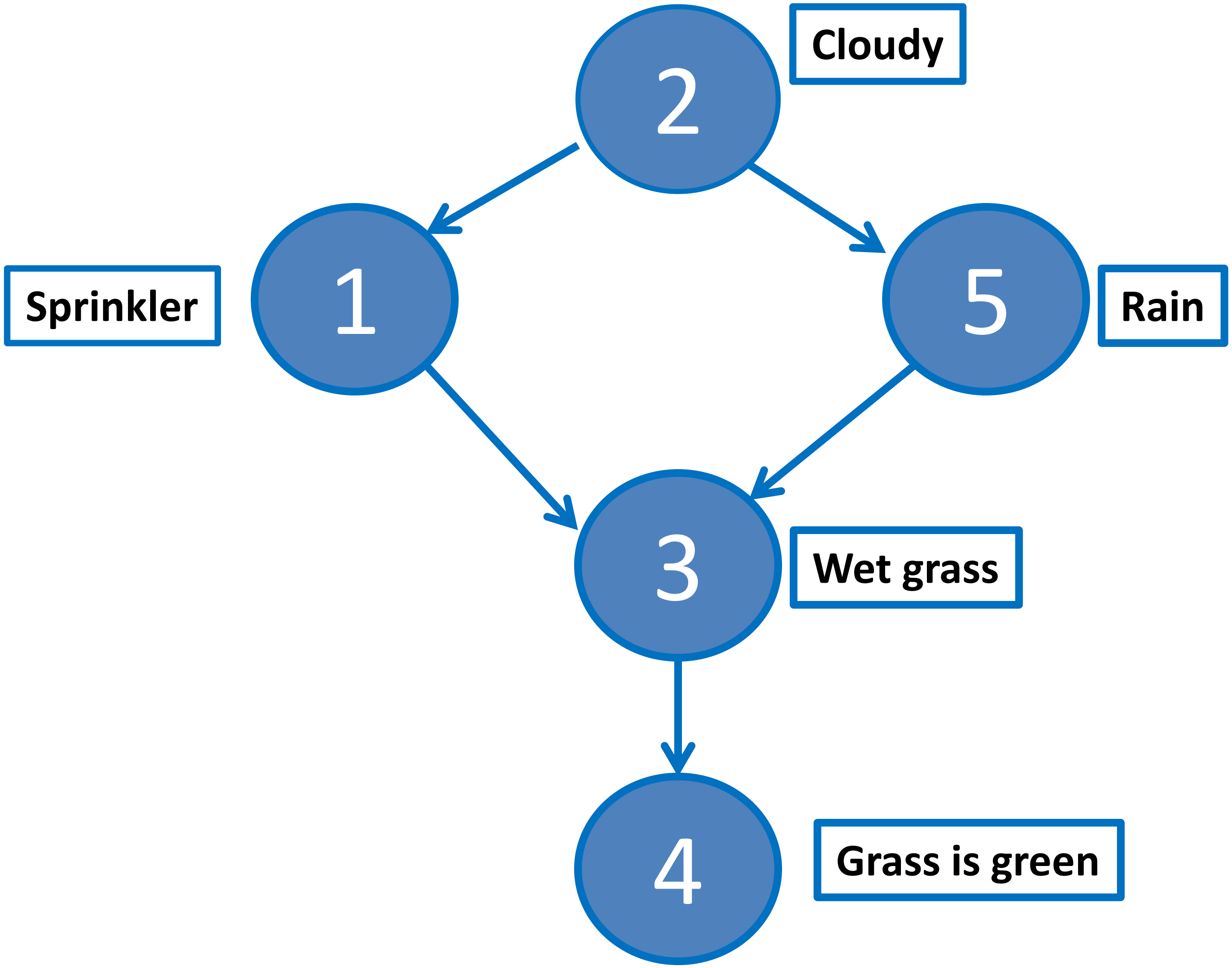
- Most of the queries one may ask involve **evidence**
  - Evidence $\mathbf{x}_v$ is an assignment of values to a set $\mathbf{X}_v$ of nodes in the GM over varialbe set $\mathbf{X}=\{X_1, X_2, \ldots, X_n\}$
  - Without loss of generality $\mathbf{X}_v=\{X_{k+1}, \ldots, X_n\}$,
  - Write $\mathbf{X}_H=\mathbf{X}\backslash\mathbf{X}_v$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

- Simplest query: compute probability of evidence

$$P(\mathbf{x}_v) = \sum_{\mathbf{x}_H} P(\mathbf{X}_H, , \mathbf{X}_v) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, \ldots, x_k, \mathbf{x}_v)$$

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

Assess Conditional Independence of Two Nodes in Bayesian Networks



P(Rain = T) =?

Inferential Query 2: Conditional Probability

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

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

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

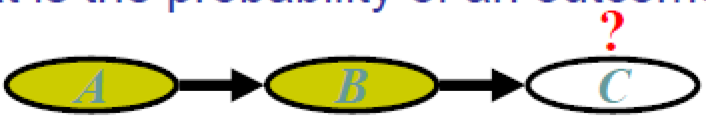
- We usually query a subset $\mathbf{Y}$ of all hidden variables $\mathbf{X_H}=\{\mathbf{Y},\mathbf{Z}\}$ and "don't care" about the remaining, $\mathbf{Z}$:

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

  - the process of summing out the "don't care" variables z is called **marginalization**, and the resulting $P(\mathbf{Y}|\mathbf{x_v})$ is called a **marginal** prob.

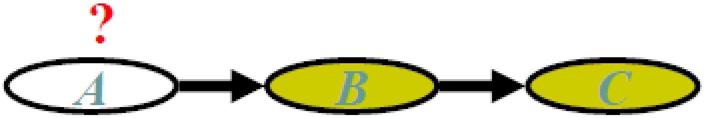
Applications of a posterior belief

- **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

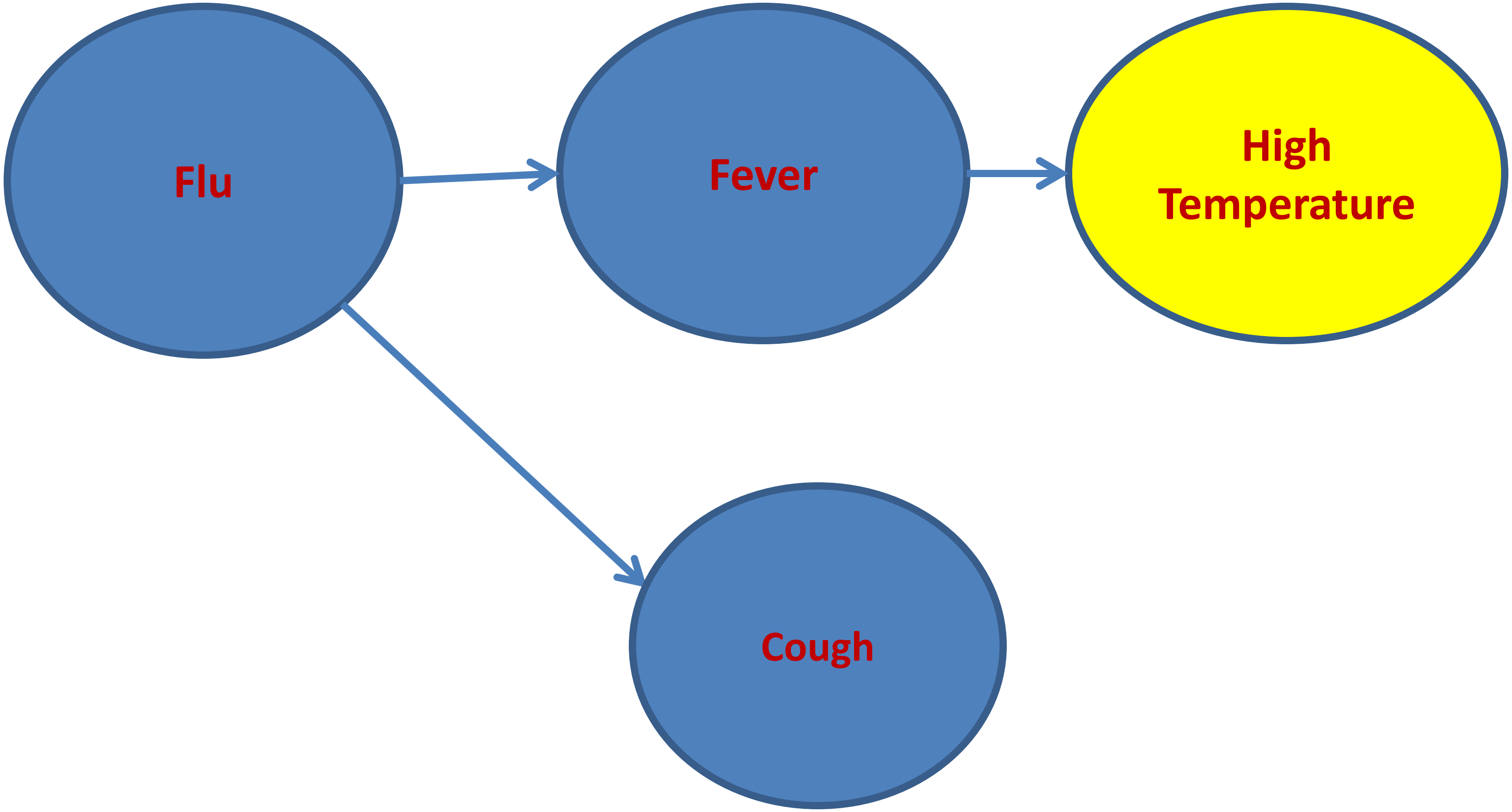


  - the query node an ancestor of the evidence

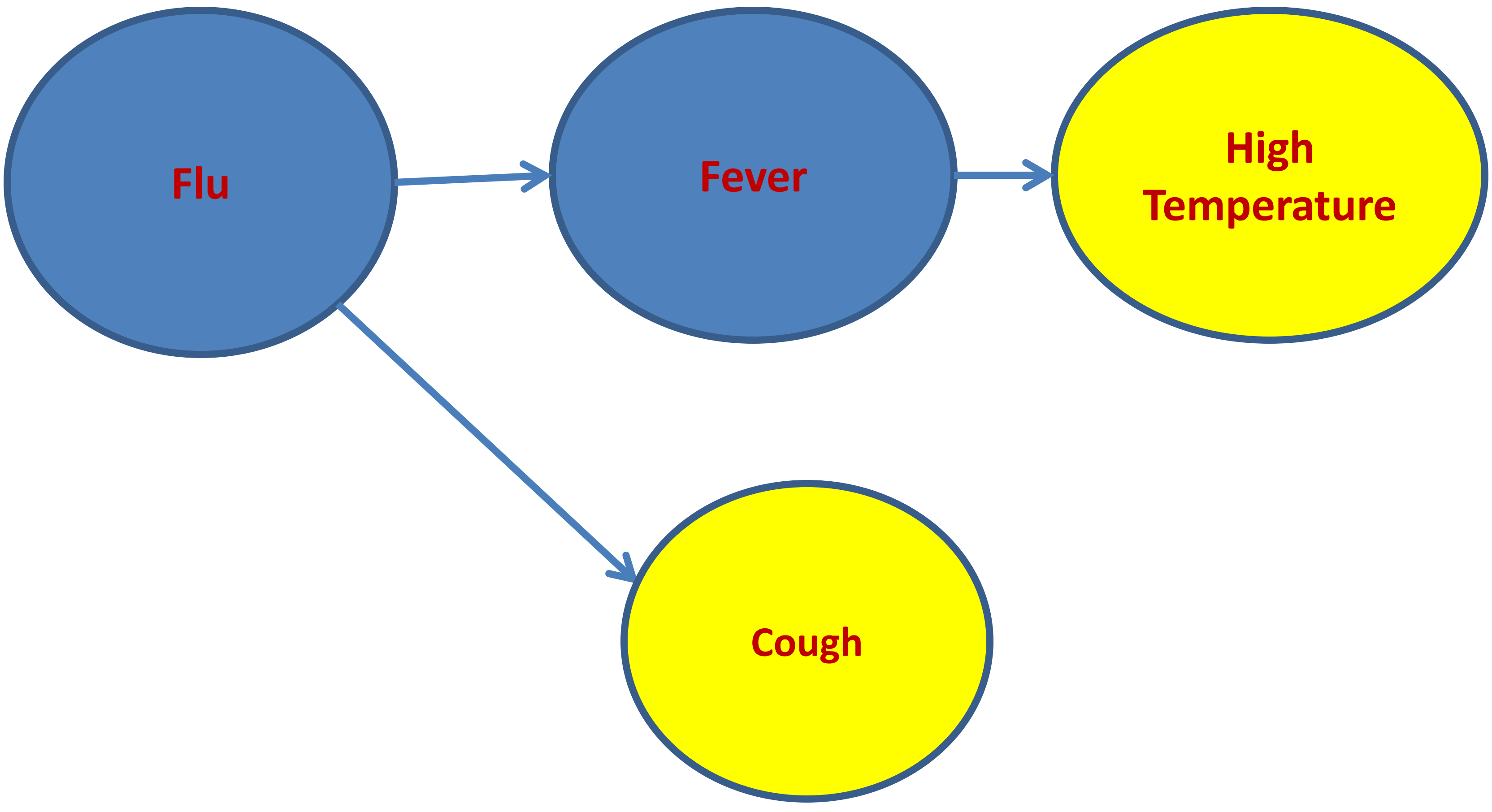
- **Learning** under partial observation
  - fill in the unobserved values under an "EM" setting (more later)

- 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

An Example



An Example – Combining Evidences



Inferential query 3: most probable assignment

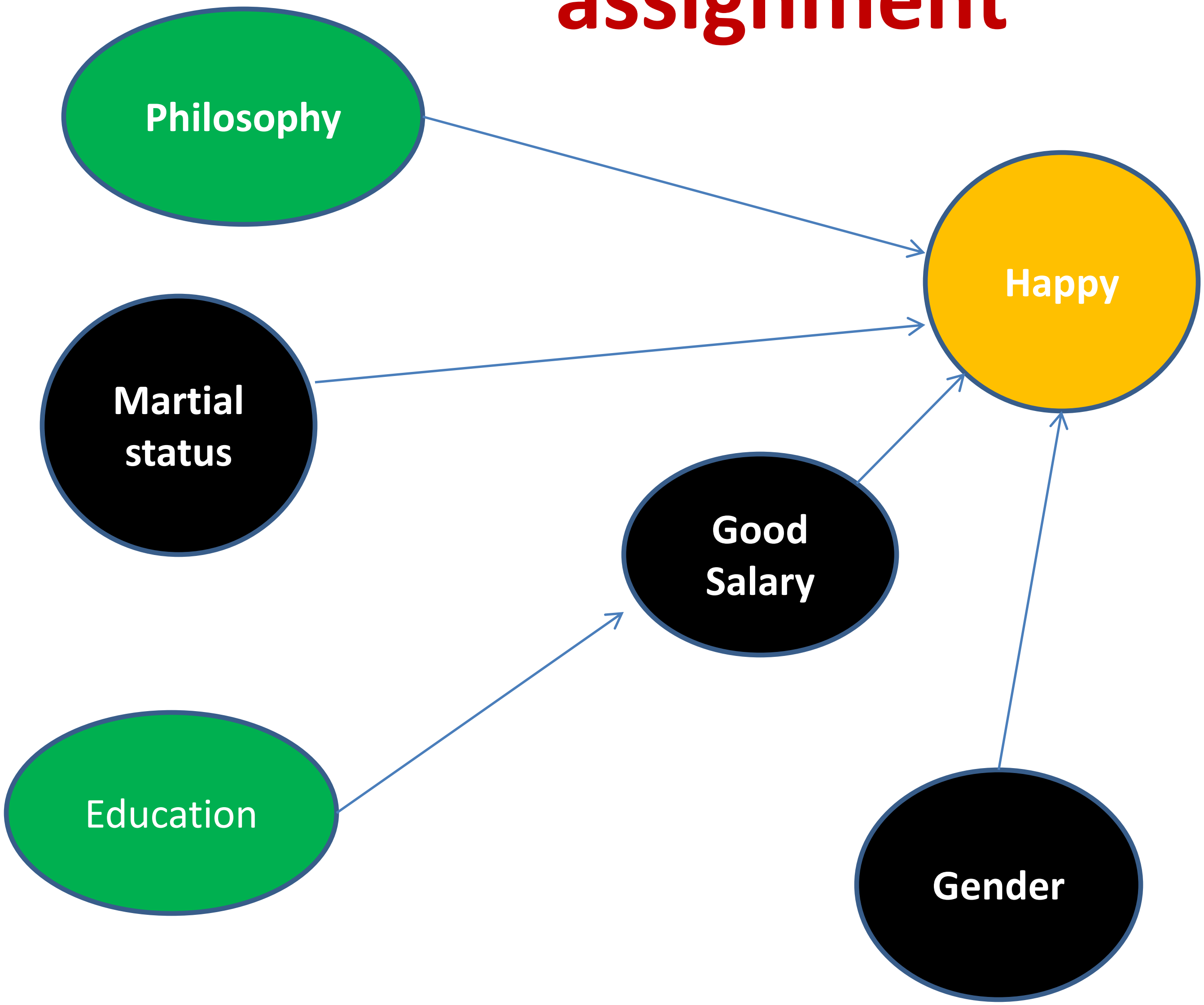
- In this query we want to find the **most probable joint assignment** (MPA) for ***some*** variables of interest

- Such reasoning is usually performed under some given evidence $\mathbf{x}_v$, and ignoring (the values of) other variables $\mathbf{Z}$:

$$\mathbf{Y}^* \mid \mathbf{x}_V = \arg\max_{\mathrm{y}} P(\mathbf{Y} \mid \mathbf{x}_V) = \arg\max_{\mathrm{y}} \sum_{\mathbf{z}} P(\mathbf{Y}, \mathbf{Z} = \mathbf{z} \mid \mathbf{x}_V)$$

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

Inferential query 3: most probable assignment



Complexity of Inference

Thm:

Computing $P(X_H = x_H | x_v)$ in an arbitrary BN 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 BNs
  - For particular families of BNs, we can have provably efficient procedures

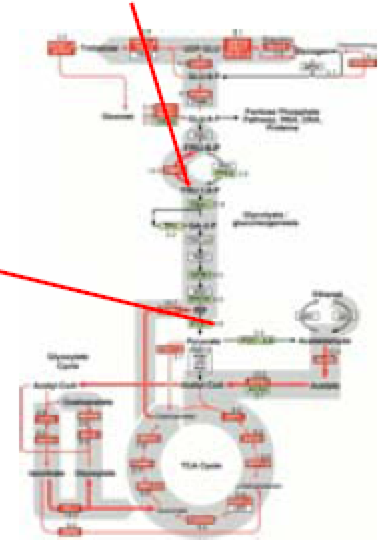
Approach to Inference

- Exact inference algorithms
  - The elimination algorithm √
  - The junction tree algorithms
- Approximate inference techniques
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods
  - Variational algorithms (will be covered in advanced ML courses)

Marginalization and Elimination

- A signal transduction pathway:

What is the likelihood that protein E is active?



- Query: $P(e)$

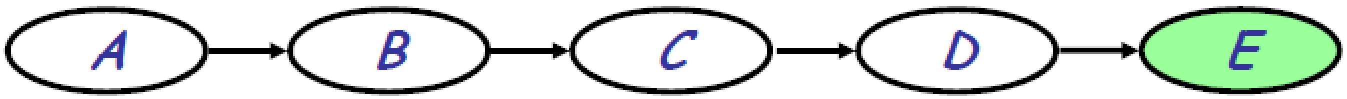
$$P(e) = \underbrace{\sum_d \sum_c \sum_b \sum_a}_{} P(a,b,c,d,e)$$

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

- By chain decomposition, we get

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

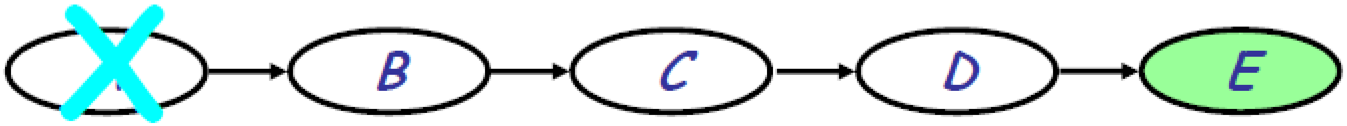
Elimination on Chains



- Rearranging terms ...

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

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

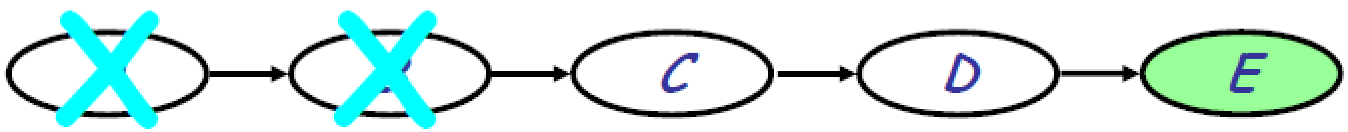


- Now we can perform innermost summation

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

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

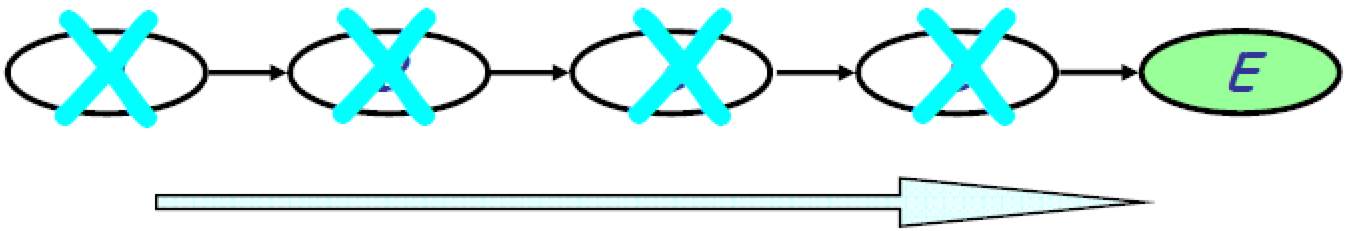
Elimination on Chains



- 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 on 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: $O(nk^2)$
  - Compare to naïve evaluation that sums over joint values of n-1 variables $O(k^n)$

Inference on General BN via Variable Elimination

General idea:

- Write query in the form

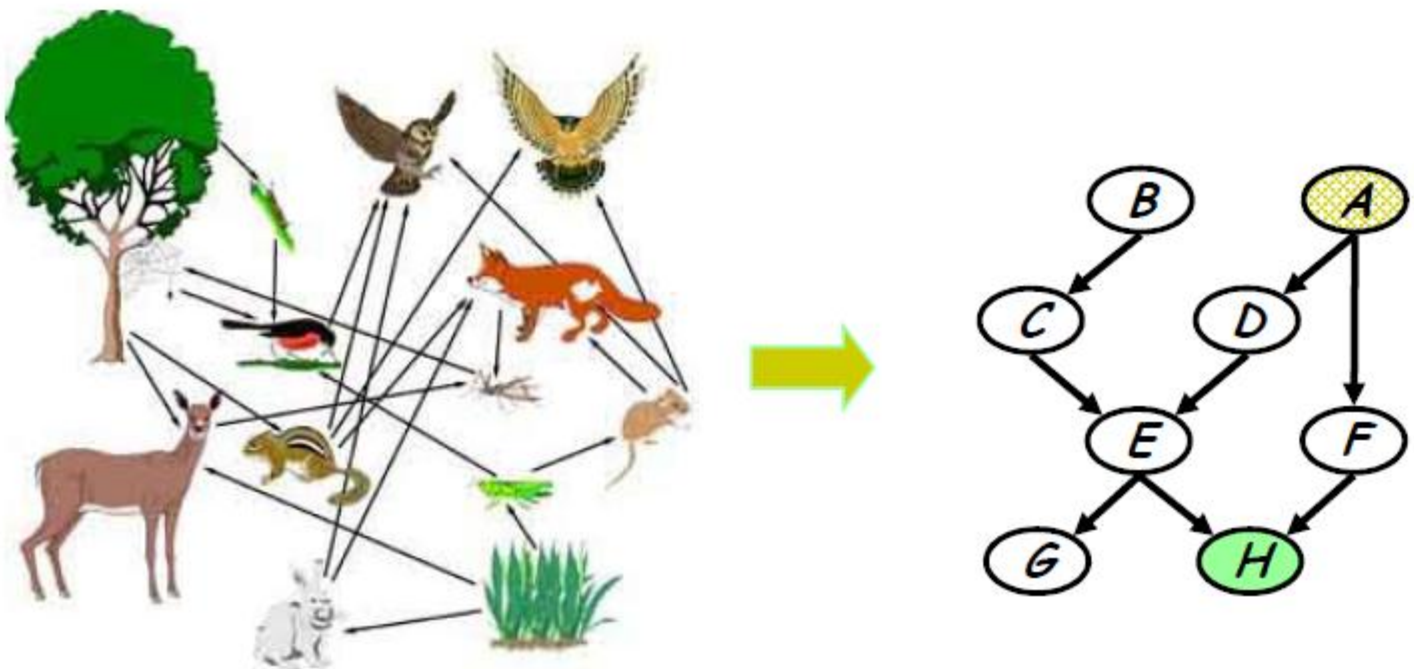
$$P(X_1, \boldsymbol{e}) = \sum_{x_n} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i \mid pa_i)$$

  - 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

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

A more complex network

A food web

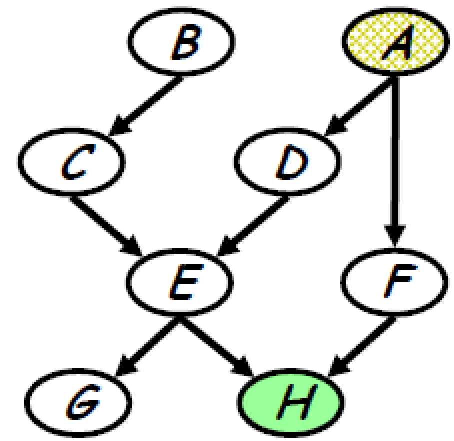


What is the probability that hawks are leaving given that the grass condition is poor?

Example: Variable Elimination

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

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



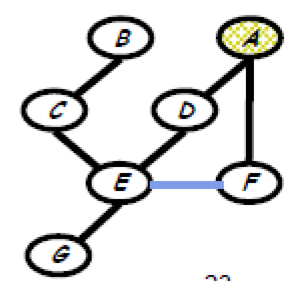
- Choose an elimination order: *H,G,F,E,D,C,B*

- 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:

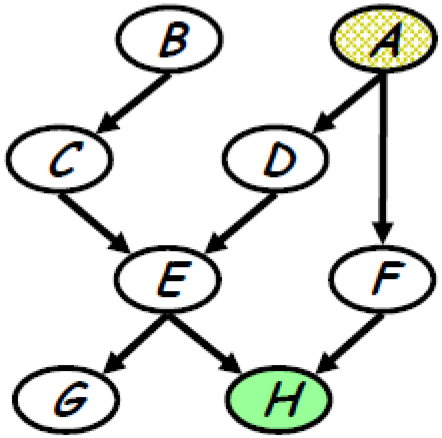
$$m_h(e,f) = \sum_h p(h \mid e,f)\delta(h = \tilde{h})$$



Example: Variable Elimination

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

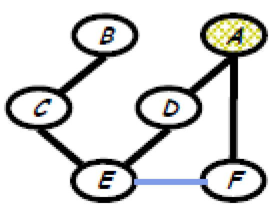
$$P(a)P(b)P(c\,|\,b)P(d\,|\,a)P(e\,|\,c,d)P(f\,|\,a)P(g\,|\,e)P(h\,|\,e,f)$$
$$\Rightarrow P(a)P(b)P(c\,|\,b)P(d\,|\,a)P(e\,|\,c,d)P(f\,|\,a)\underline{P(g\,|\,e)}m_h(e,f)$$



- Step 2: Eliminate G
  - compute

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

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

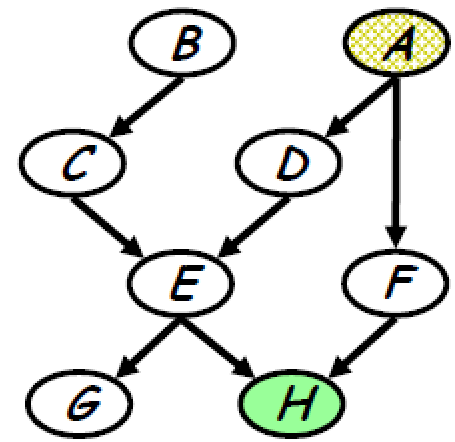


Example: Variable Elimination

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

- Initial factors:

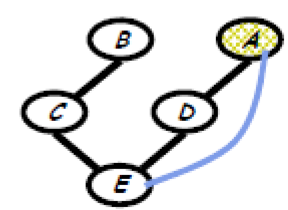
$$P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)P(h \mid e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)P(f \mid a)P(g \mid e)m_h(e,f)$$
$$\Rightarrow P(a)P(b)P(c \mid b)P(d \mid a)P(e \mid c,d)\underline{P(f \mid a)m_h(e,f)}$$



- Step 3: Eliminate F
  - compute

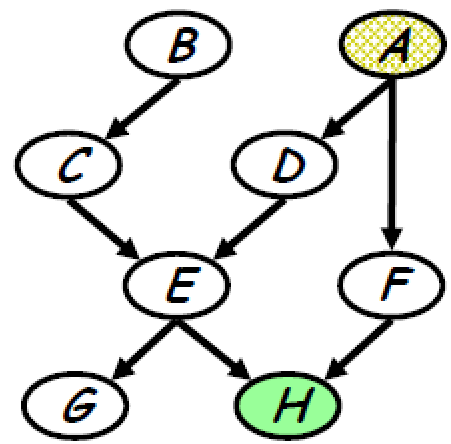
$$m_f(e,a) = \sum_f p(f \mid a)m_h(e,f)$$

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



Example: Variable Elimination

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



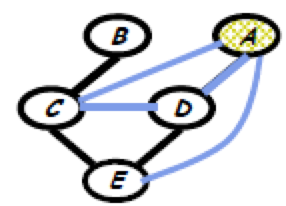
- Initial factors:

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

- Step 4: Eliminate *E*
  - compute

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

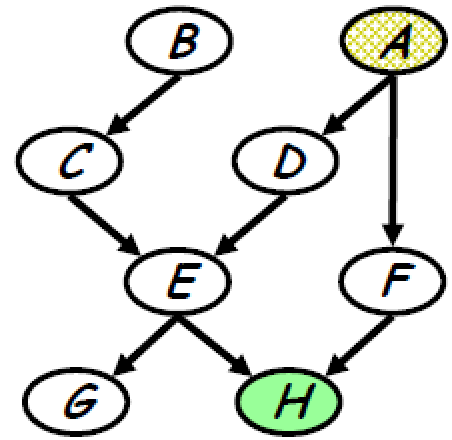
$$\Rightarrow P(a)P(b)P(c|b)P(d|a)\underline{m_e(a,c,d)}$$



Example: Variable Elimination

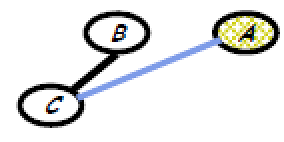
- Query: $P(B \mid h)$
  - Need to eliminate: B, C, D

- 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 elimінate: *B*, *C*
- Initial factors:

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

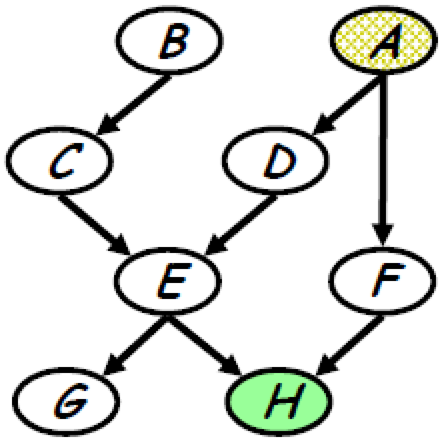
$$\Rightarrow P(a)P(b)P(c\mid d)P(d\mid a)P(e\mid c,d)P(f\mid a)P(g\mid e)m_h(e,f)$$

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

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

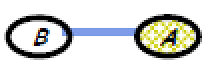
$$\Rightarrow P(a)P(b)P(c\mid d)P(d\mid a)m_e(a,c,d)$$

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



- Step 6: Eliminate *C*
  - compute

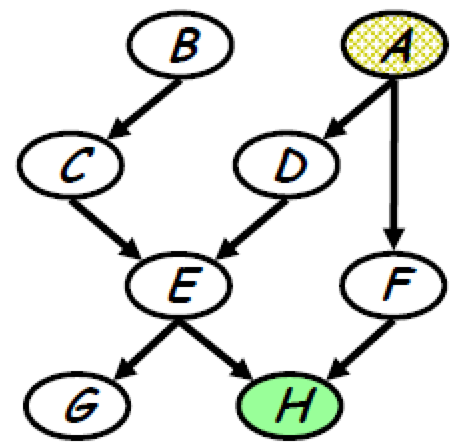
$$m_c(a,b) = \sum_c p(c\mid b)m_d(a,c)$$



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

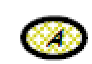
Example: Variable Elimination

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



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

- Step 7: Eliminate *B*



  - compute

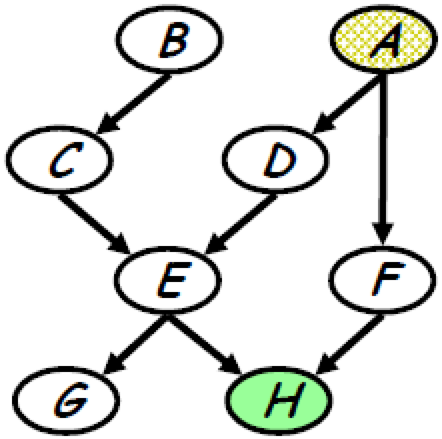
$$m_b(a) = \sum_b p(b)m_c(a,b)$$

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

Example: Variable Elimination

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

- Initial factors:



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

- Step 8: Wrap-up $\quad p(a,\tilde{h}) = p(a)m_b(a), \quad p(\tilde{h}) = \sum_a p(a)m_b(a)$

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

Complexity of Variable Elimination

- Suppose in one elimination step we compute

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

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

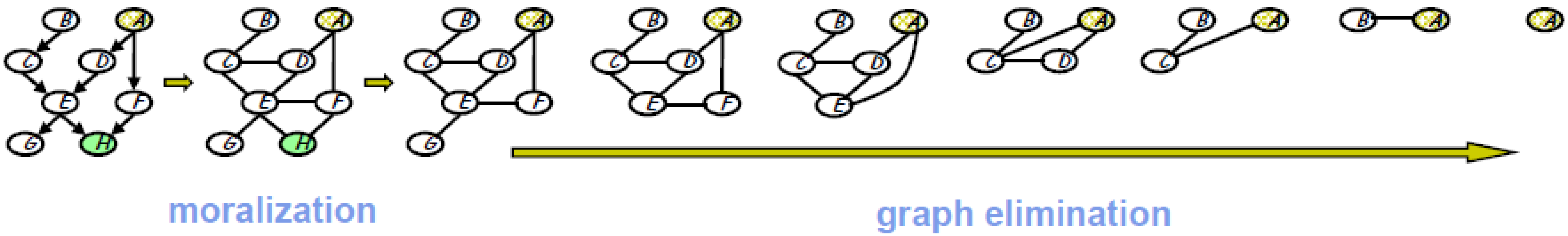
This requires

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

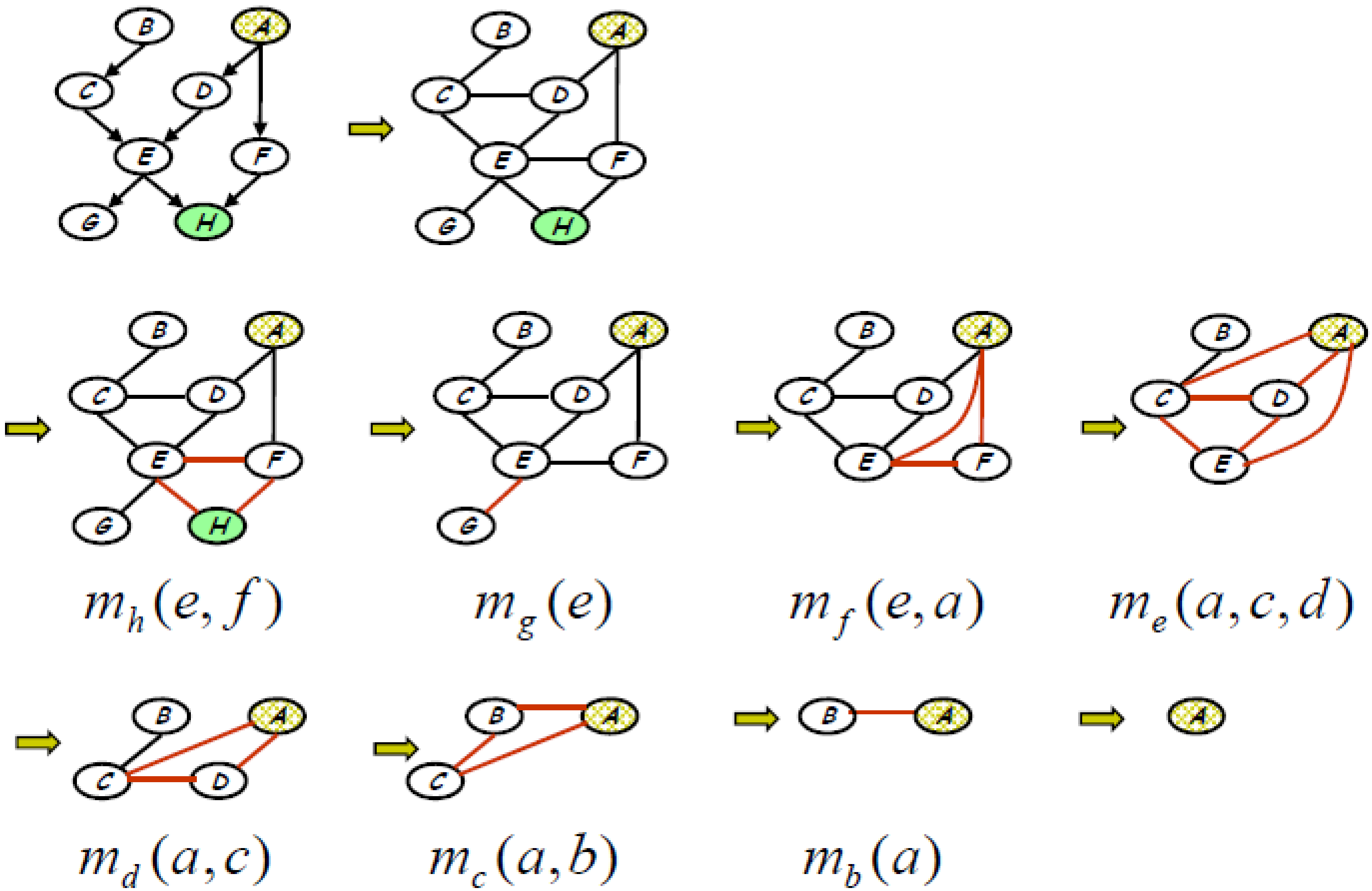
Complexity is **exponential** in number of variables in the intermediate factor

Understanding Variable Elimination

- A graph elimination algorithm

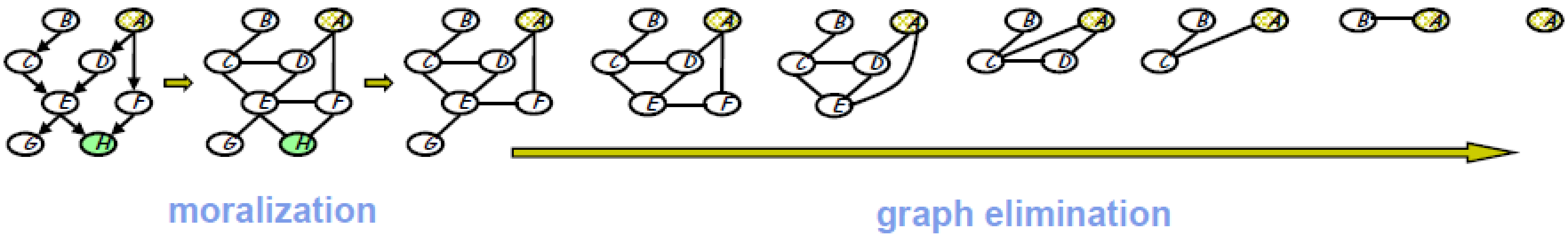


Elimination Cliques



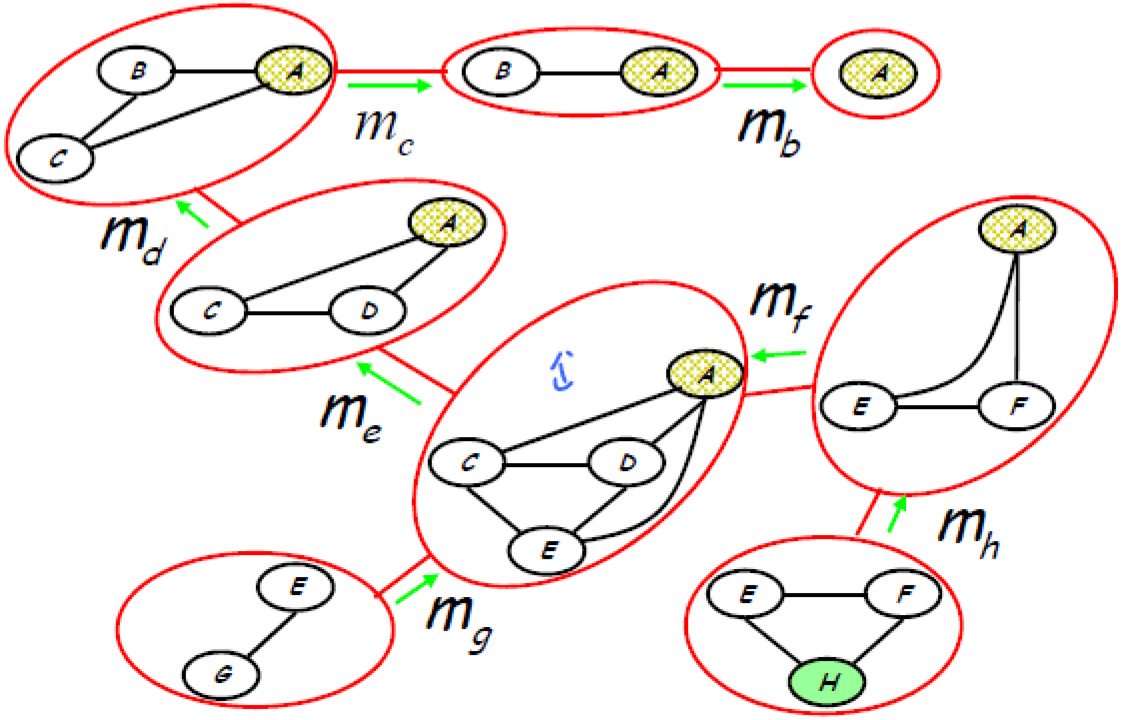
Understanding Variable Elimination

- 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

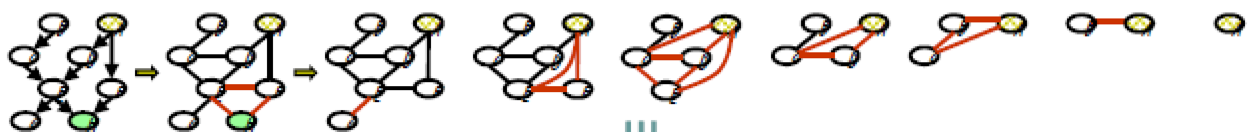
A Clique Tree



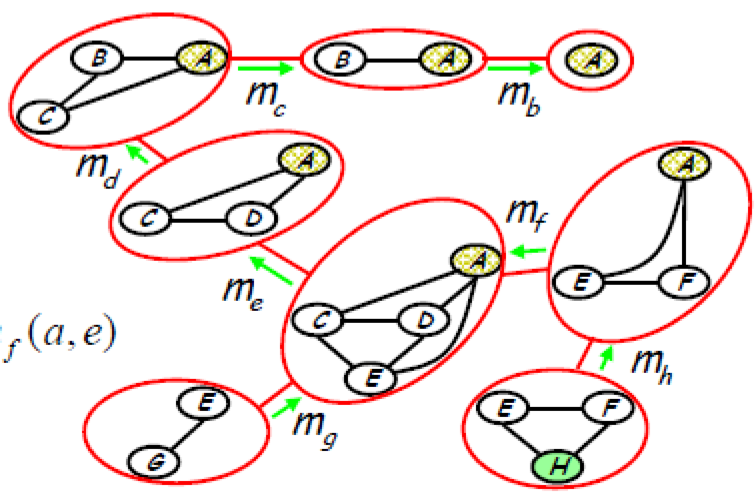
$$m_e(a,c,d)$$
$$= \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

From Elimination to Message Passing

- 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**



$\equiv$

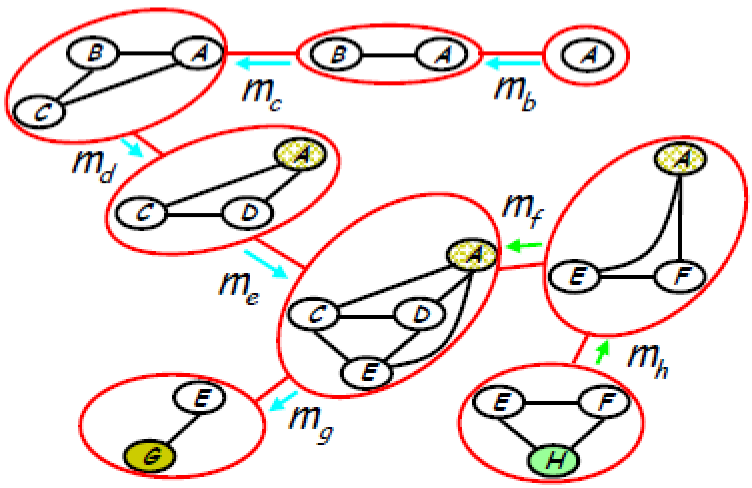


$$m_e(a,c,d) = \sum_e p(e \mid c,d) m_g(e) m_f(a,e)$$

- Messages can be reused

From Elimination to Message Passing

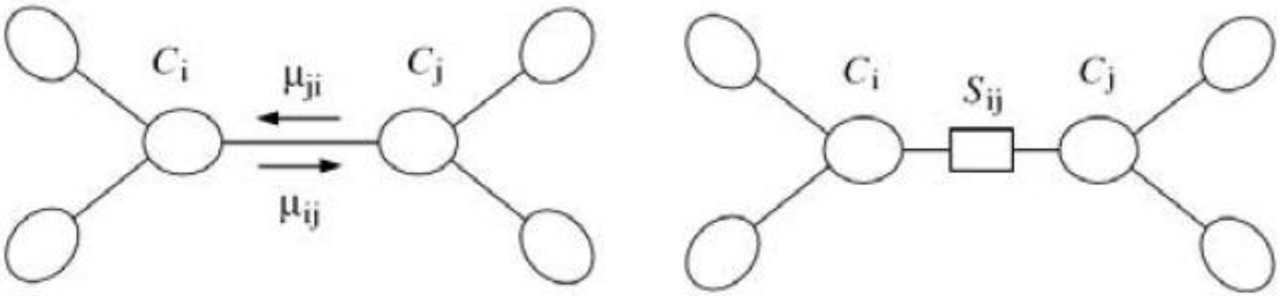
- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination $\equiv$ message passing on a **clique tree**
  - **Another query ...**



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

The Junction Tree Algorithm

- Shafer-Shenoy algorithm



(a) (b)

  - Message from clique *i* to clique *j* :

$$\mu_{i\to j} = \sum_{C_i \setminus S_{ij}} \psi_{C_i} \prod_{k\neq j} \mu_{k\to i}(S_{ki})$$

  - Clique marginal

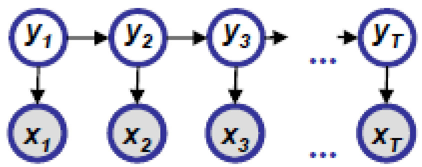
$$p(C_i) \propto \psi_{C_i} \prod_{k} \mu_{k\to i}(S_{ki})$$

The Sketch of Junction Tree Algorithm

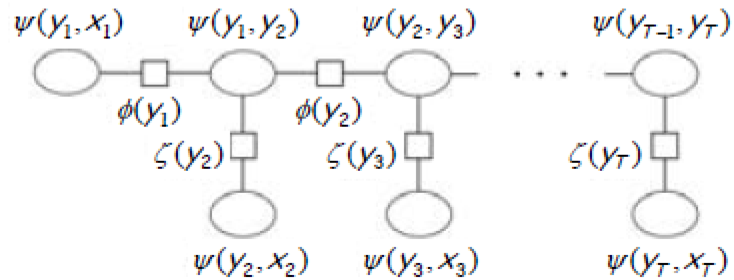
- **The algorithm**
  - Construction of junction trees --- a special **clique tree**
  - Propagation of probabilities --- a message-passing protocol
- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: 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
  - Forward-backward, Kalman filter, Peeling, Sum-Product ...

A Junction Tree Algorithm for HMM

- A junction tree for the HMM



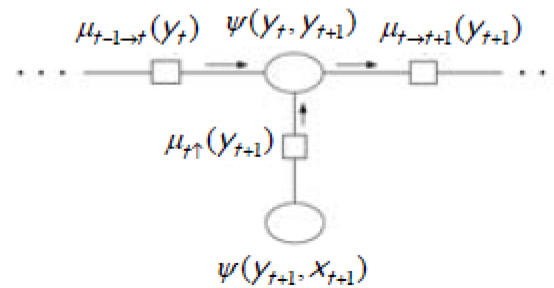
$\Rightarrow$



- Rightward pass

$$\mu_{t\to t+1}(y_{t+1}) = \sum_{y_t} \psi(y_t, y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_t} p(y_{t+1} \mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1} \mid y_{t+1})$$
$$= p(x_{t+1} \mid y_{t+1})\sum_{y_t} a_{y_t, y_{t+1}}\mu_{t-1\to t}(y_t)$$

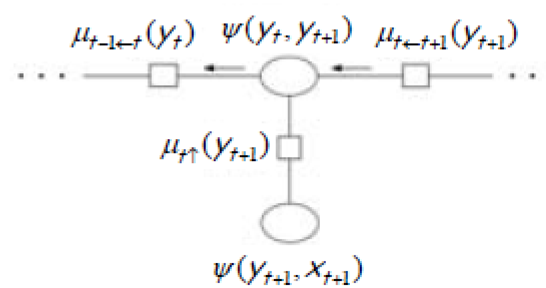
  - This is exactly the *forward algorithm*!



- Leftward pass …

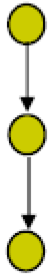
$$\mu_{t-1\leftarrow t}(y_t) = \sum_{y_{t+1}} \psi(y_t, y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$
$$= \sum_{y_{t+1}} p(y_{t+1} \mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1} \mid y_{t+1})$$

  - This is exactly the *backward algorithm*!



Summary

- Represent dependency structure with a directed acyclic graph
  - Node <-> random variable
  - Edges encode dependencies
    - Absence of edge -> conditional independence
  - Plate representation
  - A BN 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
- Support efficient inference and learning