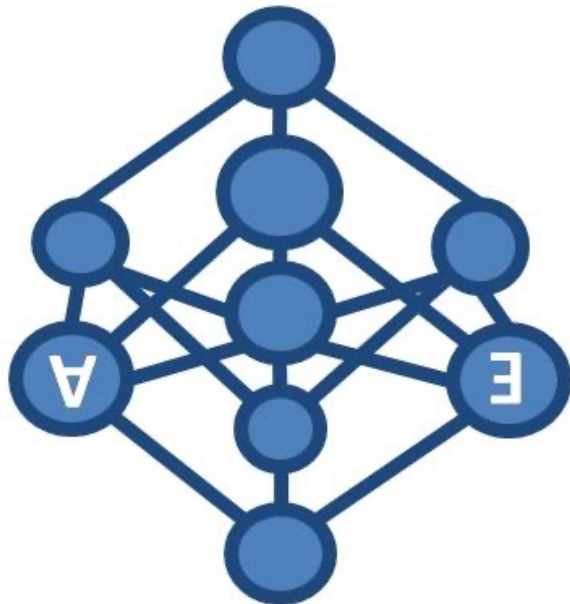


# Probabilistic Graphical Models\*

## Bayesian Networks - Inference



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



\*Thanks to Bert Huang, Carlos Guestrin, Pedro Domingos and many others for making their slides publically available



# What's next

- So far, variable elimination for “efficient” inference on conditional probability queries.
- Now:
  - Other types of inference
  - Hardness result of inference

# So far: A-Posteriori Belief

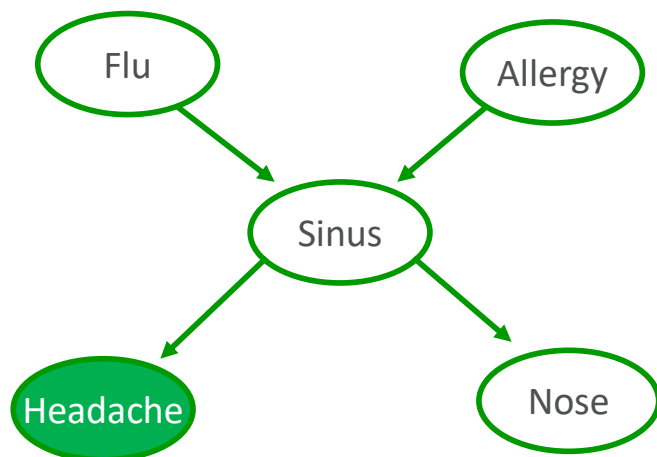
This query is useful in many cases:

- **Prediction:** what is the probability of an outcome given the starting condition
  - Target is a descendent of the evidence
- **Diagnosis:** what is the probability of disease/fault given symptoms
  - Target is an ancestor of the evidence
- So, the direction between variables does not restrict the directions of the queries. Probabilistic inference can combine evidence from all parts of the network

# Abductive Inference in BNs

So far, we have considered inference problems where the goal is to obtain **posterior probabilities for variables given evidence**.

In abductive inference it is to find the **configuration of a set of variables (hypothesis) which will best explain the evidence**.



What would count as the best explanation of an headache ( $H=t$ )?

A configuration of all the other variables?

A subset of them?

# Abductive Inference in BNs

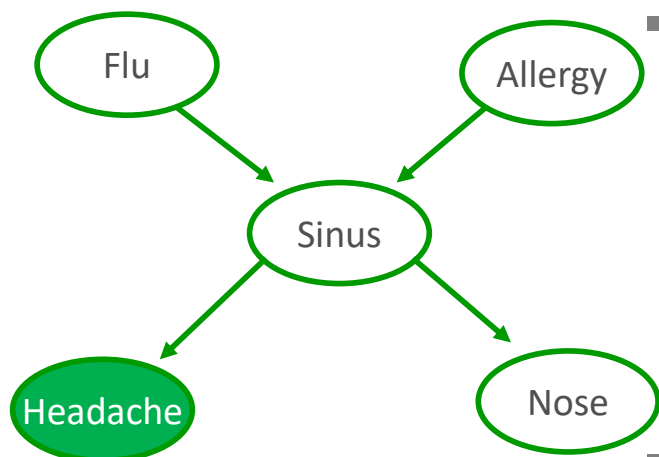
There are two types of abductive inference in BNs:

- **MPE (Most Probable Explanation)** - the most probable configuration of *all variables* in the BN given evidence
- **MAP (Maximum A Posteriori)** - the most probable configuration of *a subset of variables* in the BN given evidence

Note 1: In general the MPE cannot be found by taking the most probable configuration of nodes individually!

Note 2: And the MAP cannot be found by taking the projection of the MPE onto the explanation set!

# Abductive Inference in BNs



*some times called maximum a posteriori (MAP)*

## Most probable explanation (MPE)

- Most likely assignment to all hidden vars given evidence

$$\max_{f,a,s,n} P(F = f, A = a, S = s, N = n \mid H = t)$$

## Maximum a posteriori (MAP)

- Most likely assignment to some var(s) given evidence

$$\max_a P(A = a \mid H = t)$$

$$= \max_a \sum_{s,f,b} P(F = f, a, s, n \mid H = t)$$

# Why MPE and MAP?

We can use MPE and MAP for

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

# Are MPE and MAP Consistent?



$$P(S=t)=0.4$$

$$P(S=f)=0.6$$

$P(N S)$	$N=t$	$N=f$
$S=t$	0.9	0.1
$S=f$	0.5	0.5

**MPE and MAP are not consistent!!**

## ■ Most probable explanation (MPE)

- Most likely assignment to all hidden variables given evidence
- $S=t, N=t: 0.4 \cdot 0.9 = 0.36$
- $S=f, N=t: 0.6 \cdot 0.5 = 0.2$
- So, we should assume to have a sinus and a running nose

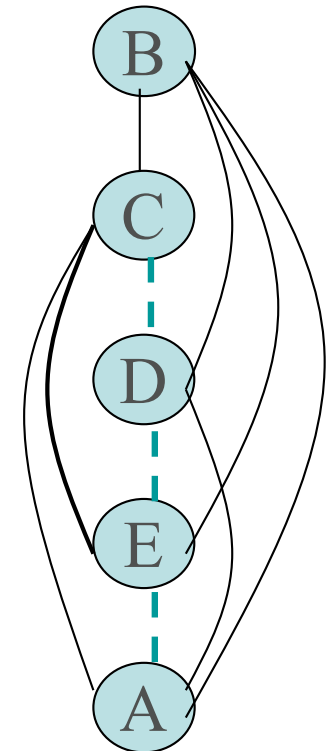
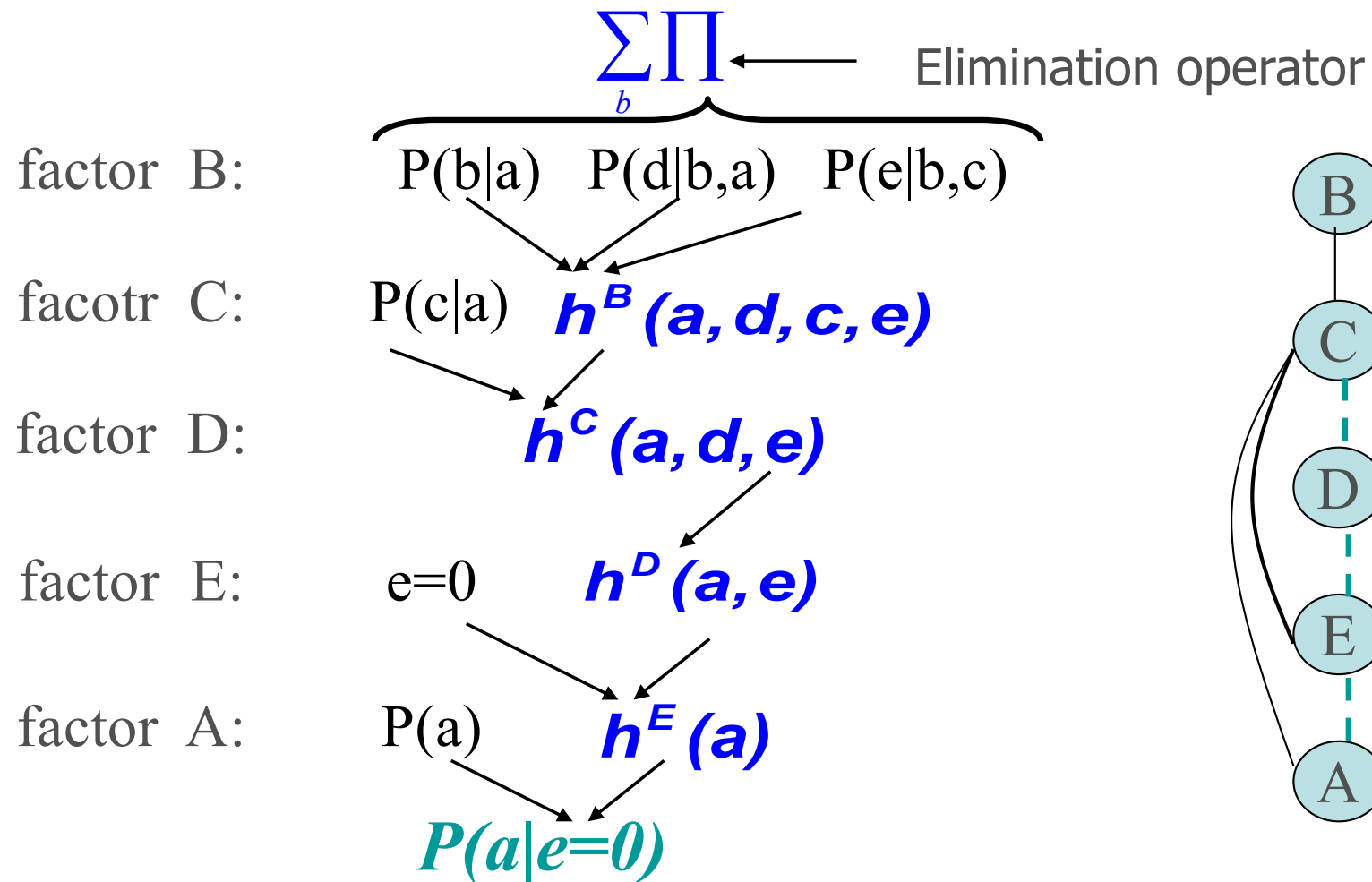
## ■ Maximum a posteriori (MAP)

- Most likely assignment to some  $\text{var}(s)$  given evidence
- According to the numbers,  $P(S=f)$  is higher, so a priori we do **not** have a sinus.





# Finding MPE



# Finding MPE

$$MPE = \max_{a,e,d,c,b} P(a)P(c|a)P(b|a)P(d|a,b)P(e|b,c)$$

$\sum$  is replaced by *max* :

$$\max_b \prod \leftarrow \text{Elimination operator}$$

factor B:

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

factor C:

$$P(c|a) \quad h^B(a, d, c, e)$$

factor D:

$$h^C(a, d, e)$$

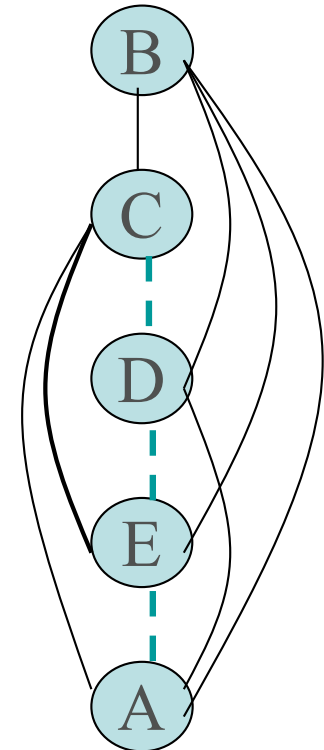
factor E:

$$e=0 \quad h^D(a, e)$$

factor A:

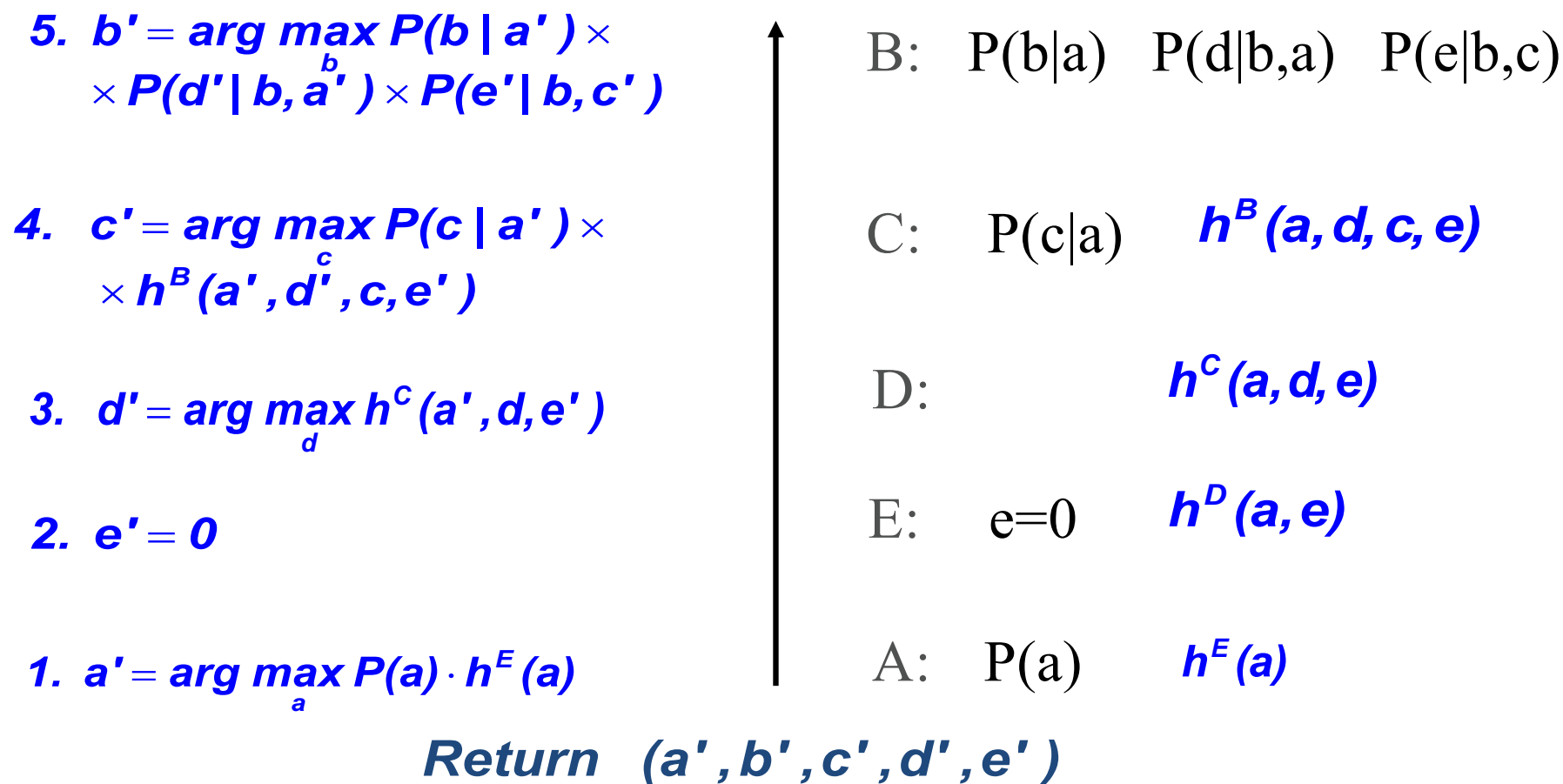
$$P(a) \quad h^E(a)$$

$$P(a|e=0)$$



# Generating the MPE-tuple

**Two passes algorithm:  
(Top-Down) Max Probs (Bottom-Up) Max Configuration**



# So far

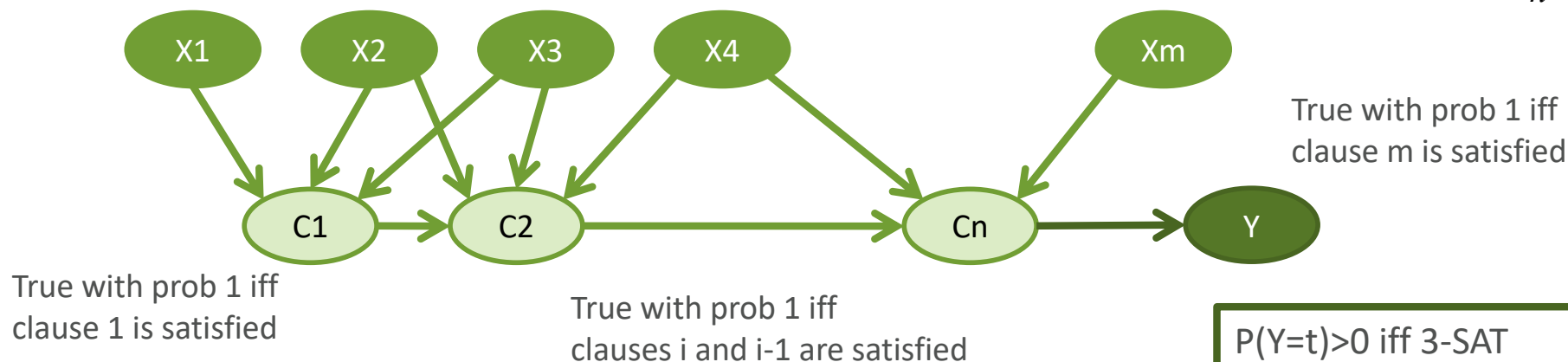
- Variable elimination for “efficient” inference on conditional probability queries.
- Now, general hardness result of inference

# Complexity of conditional probability queries

- How hard is it to compute  $P(X | \mathbf{E}=\mathbf{e})$ ?
- Consider a reduction to 3-SAT with empty evidence  $E$
- Does a satisfying assignment exist?

$$\underbrace{(\bar{X}_1 \vee X_2 \vee X_3)}_{C_1} \wedge \underbrace{(\bar{X}_2 \vee X_3 \vee X_4)}_{C_2} \wedge \dots \wedge \underbrace{\dots}_{C_n}$$

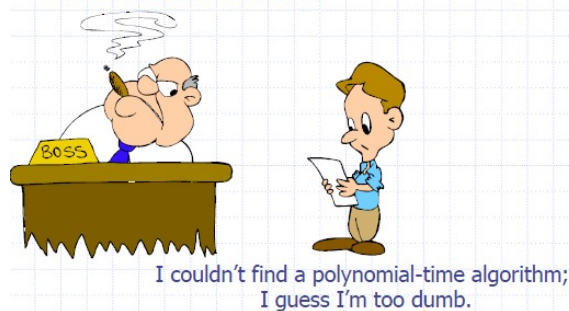
0.5/0.5 prior



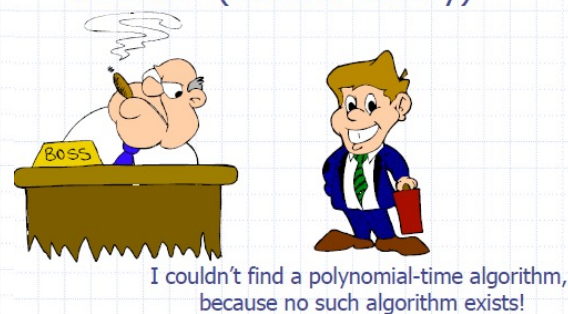
$P(Y=t) > 0$  iff 3-SAT formula is satisfiable



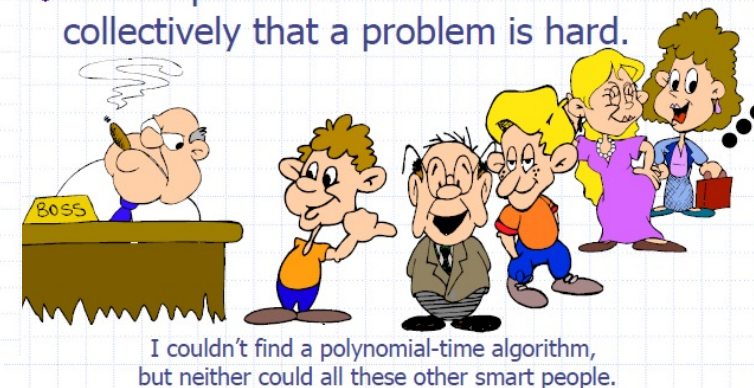
◆ What to do when we find a problem that looks hard...



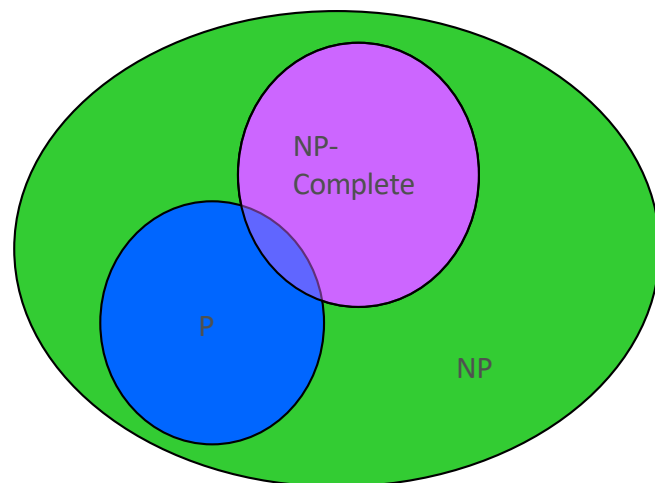
◆ Sometimes we can prove a strong lower bound... (but not usually)



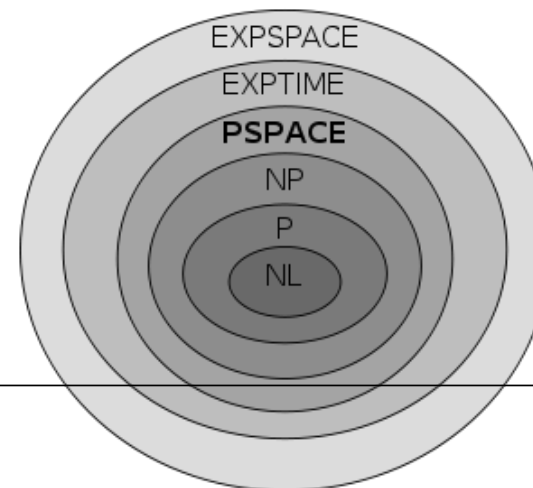
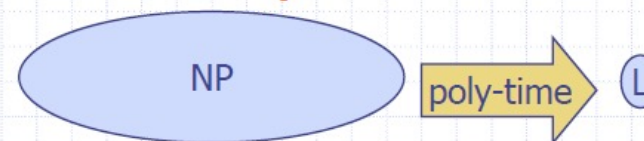
◆ NP-completeness let's us show collectively that a problem is hard.



- $P = \{ L \mid L \text{ is accepted by a deterministic Turing Machine in polynomial time} \}$
- $NP = \{ L \mid L \text{ is accepted by a non-deterministic Turing Machine in polynomial time} \}$



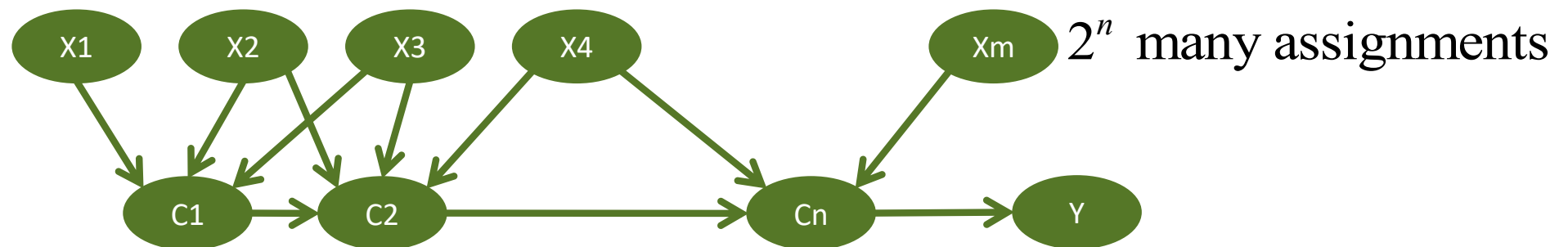
- ◆ A problem (language)  $L$  is **NP-hard** if every problem in NP can be reduced to  $L$  in polynomial time.
- ◆ That is, for each language  $M$  in NP, we can take an input  $x$  for  $M$ , **transform** it in polynomial time to an input  $x'$  for  $L$  such that  $x$  is in  $M$  if and only if  $x'$  is in  $L$ .
- ◆  $L$  is **NP-complete** if it's in NP and is NP-hard.



# Complexity of conditional probability queries

- How hard is it to compute  $P(X | \mathbf{E}=\mathbf{e})$ ?
  - At least NP-hard, but even harder!
  - #P problems such as counting the number of satisfiable configurations (model counting)

0.5/0.5 prior



$$p(Y = t) = \frac{\# \text{ sat assignment}}{2^n}$$

# Hardness - Notes

- We used deterministic relations in our construction
- The same construction works if we use  $(1-\varepsilon, \varepsilon)$  instead of  $(1,0)$  in each gate for any  $\varepsilon < 0.5$
- **Hardness does not mean we cannot solve inference**
- It implies that we cannot find a general procedure that works efficiently for all networks
- For particular families of networks, we can have provably efficient procedure



# What you need to know about inference thus far

## ■ Types of queries

- probabilistic inference
- most probable explanation (MPE)
- maximum a posteriori (MAP)
  - MPE and MAP are truly different (don't give the same answer)

## ■ Hardness of inference

- Exact and approximate inference are NP-hard
- MPE is NP-complete
- MAP is much harder as we solve the model counting problem

# What's next

- Understanding complexity of variable elimination in more detail
  - Variable elimination as graph transformation
  - Will lead to junction-tree algorithm
  - Will provide some background on MRFs and (loopy) belief propagation

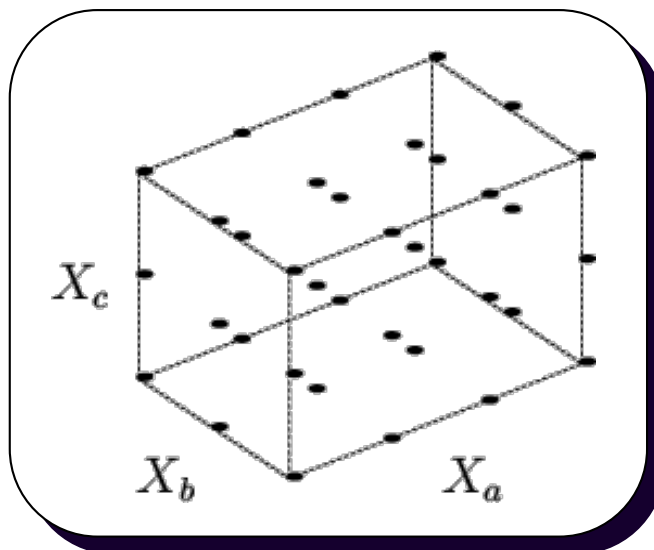
# Recap Potentials

A **potential**  $f_A$  over a set of variables  $A$  is a function that maps each configuration into a **non-negative real number**.

$$\text{dom} f_A = A \quad (\text{domain of } f_A)$$

**Examples:** Conditional probability distribution and joint probability distributions are special cases of potentials

# Potentials



Ex: A potential  $\phi_{A,B,C}$  over the set of variables  $\{A,B,C\}$ . A has four states, and B and C has three states.  **$\text{domf}_{A,B,C} = \{A,B,C\}$**

# CPTs as Potentials

Potentials: We can represent a CPT in this format...

$D$	$E$	$P(F)$
T	T	0.8
T	F	0.5
F	T	0.2
F	F	0.7



$d$	$e$	$f$	.8
$d$	$e$	$\neg f$	.2
$d$	$\neg e$	$f$	.5
$d$	$\neg e$	$\neg f$	.5
$\neg d$	$e$	$f$	.2
$\neg d$	$e$	$\neg f$	.8
$\neg d$	$\neg e$	$f$	.7
$\neg d$	$\neg e$	$\neg f$	.3

# Multiplying Potentials

- Domain of (variables in) result is the union of domains of input potentials
- For each cell in result, multiply all input cells that agree on variable settings

$$\begin{array}{|c|c|c|} \hline a & b & .1 \\ \hline a & \neg b & .2 \\ \hline \neg a & b & .5 \\ \hline \neg a & \neg b & .8 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline b & c & .2 \\ \hline b & \neg c & .4 \\ \hline \neg b & c & .3 \\ \hline \neg b & \neg c & .5 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline a & b & c & .02 \\ \hline a & b & \neg c & .04 \\ \hline a & \neg b & c & .06 \\ \hline a & \neg b & \neg c & .10 \\ \hline \neg a & b & c & .10 \\ \hline \neg a & b & \neg c & .20 \\ \hline \neg a & \neg b & c & .24 \\ \hline \neg a & \neg b & \neg c & .40 \\ \hline \end{array}$$
  

$$\begin{array}{|c|c|c|} \hline a & b & .1 \\ \hline a & \neg b & .5 \\ \hline \neg a & b & .4 \\ \hline \neg a & \neg b & .1 \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline a & b & .8 \\ \hline a & \neg b & .7 \\ \hline \neg a & b & .9 \\ \hline \neg a & \neg b & .8 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline a & b & .08 \\ \hline a & \neg b & .35 \\ \hline \neg a & b & .36 \\ \hline \neg a & \neg b & .08 \\ \hline \end{array}$$

# Marginalizing and Normalizing Potentials

- Can also marginalize (sum out a variable) potentials

$a$	$b$	.1
$a$	$\neg b$	.5
$\neg a$	$b$	.2
$\neg a$	$\neg b$	.7

 $\Rightarrow \sum_b$ 

$a$	.6
$\neg a$	.9

- And normalize them

$a$	$b$	.1
$a$	$\neg b$	.5
$\neg a$	$b$	.2
$\neg a$	$\neg b$	.7

 $\Rightarrow$ 

$a$	$b$	.067
$a$	$\neg b$	.333
$\neg a$	$b$	.133
$\neg a$	$\neg b$	.467

# Key Observation of VE

$$\sum_A (P_1 \times P_2) = \left( \sum_A P_1 \right) \times P_2 \quad \text{if } A \text{ is not in } P_2$$

$a$	$b$	.1
$a$	$\neg b$	.2
$\neg a$	$b$	.2
$\neg a$	$\neg b$	.3

 $\times$ 

$b$	$c$	.1
$b$	$\neg c$	.4
$\neg b$	$c$	.3
$\neg b$	$\neg c$	.1

 $=$ 

$b$	$c$	$a$	.01
$b$	$c$	$\neg a$	.02
$b$	$\neg c$	$a$	.04
$b$	$\neg c$	$\neg a$	.08
$\neg b$	$c$	$a$	.06
$\neg b$	$c$	$\neg a$	.09
$\neg b$	$\neg c$	$a$	.02
$\neg b$	$\neg c$	$\neg a$	.03

 $\Rightarrow \sum_A$ 

$b$	$c$	.03
$b$	$\neg c$	.12
$\neg b$	$c$	.15
$\neg b$	$\neg c$	.05

- can marginalize (sum out  $A$ ) before multiplying in this case, resulting in a smaller intermediate table

$a$	$b$	.1
$a$	$\neg b$	.2
$\neg a$	$b$	.2
$\neg a$	$\neg b$	.3

 $\Rightarrow \sum_A$ 

$b$	.3
$\neg b$	.5

 $\times$ 

$b$	$c$	.1
$b$	$\neg c$	.4
$\neg b$	$c$	.3
$\neg b$	$\neg c$	.1

 $=$ 

$b$	$c$	.03
$b$	$\neg c$	.12
$\neg b$	$c$	.15
$\neg b$	$\neg c$	.05



## VE using Potentials

Let  $F$  be a set of potentials (e.g. CPDs), and let  $X$  be a variable.  $X$  is eliminated from  $F$ :

1. Remove all potentials in  $F$  with  $X$  in the domain. Let  $F_X$  that set.

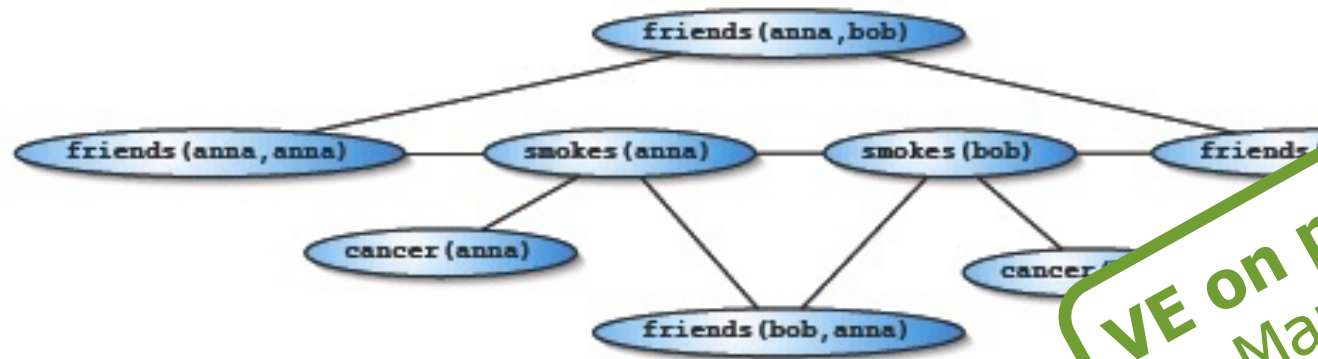
2. Calculate  $\Phi^{-X} = \sum_X \prod \Phi_X$

3. Add  $\Phi^{-X}$  to  $F$

4. Iterate

**Potential where  $X$  is not a member of the domain**

# Recap: Markov Networks / MRFs

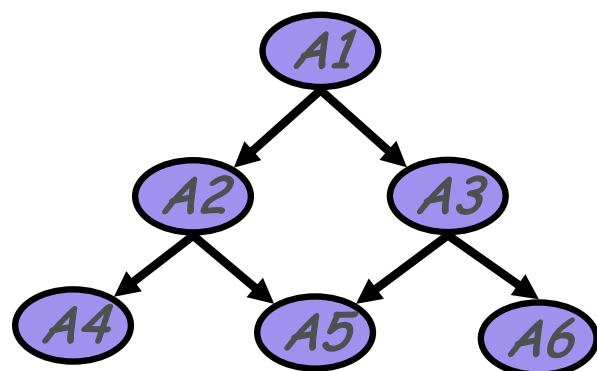


**VE on potentials** works  
for Markov networks

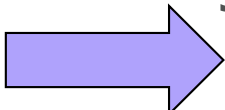
- Undirected Graphs
- Nodes = random variables  $X_1, \dots, X_n$
- Cliques = potentials ( $\sim$  local jpd)  $\phi_k$

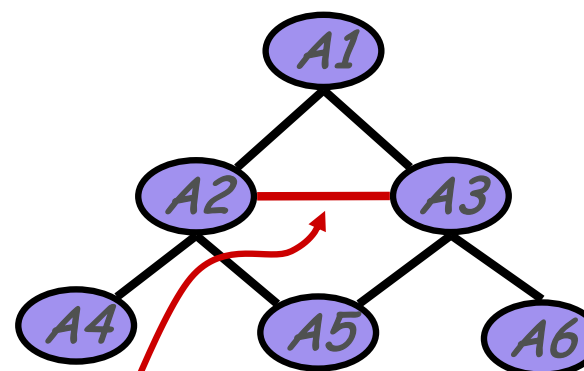
$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

# From Directed to Undirected Models: The Domain Graph



Bayesian Network

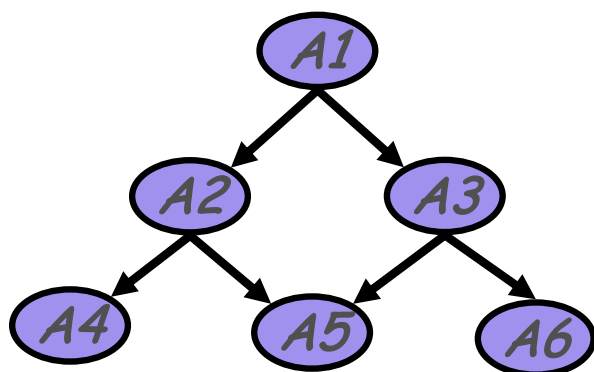
Moralizing  




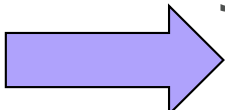
moral link  
 Domain Graph  
(Moral Graph)

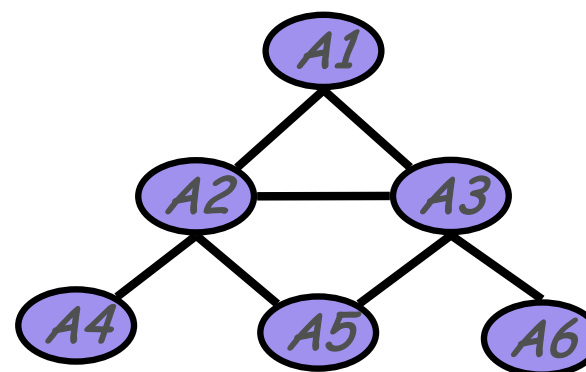
Let  $F = \{f_1, \dots, f_n\}$  be potentials over  $U = \{A_1, \dots, A_m\}$  with  $\text{dom} f_i = D_i$ . The domain graph for  $F$  is the undirected graph with variables of  $U$  as nodes and with a link between pairs of variables being members of the same  $D_i$ .

# Moralizing



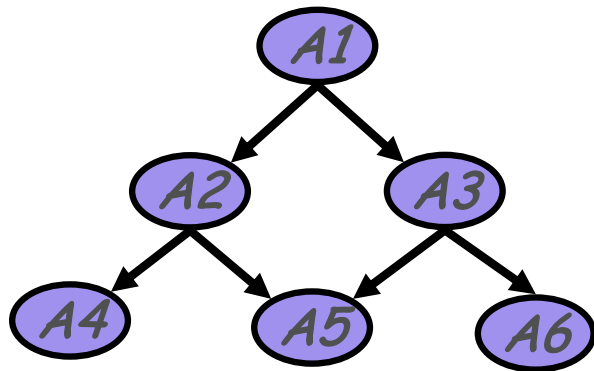
Bayesian Network

Moralizing  


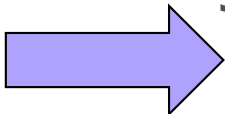


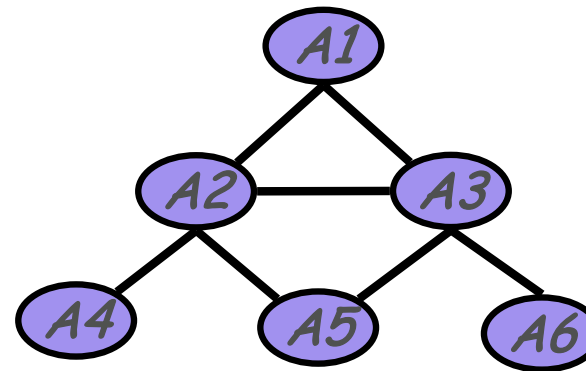
Domain Graph  
(Moral Graph)

# Moralizing

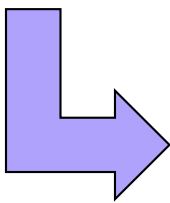


Bayesian Network

Moralizing  


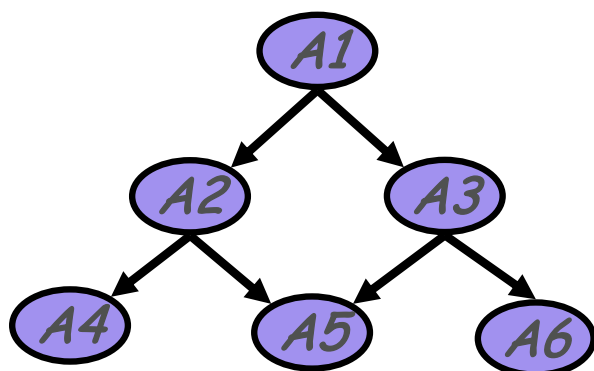


Domain Graph  
(Moral Graph)

CPDs 

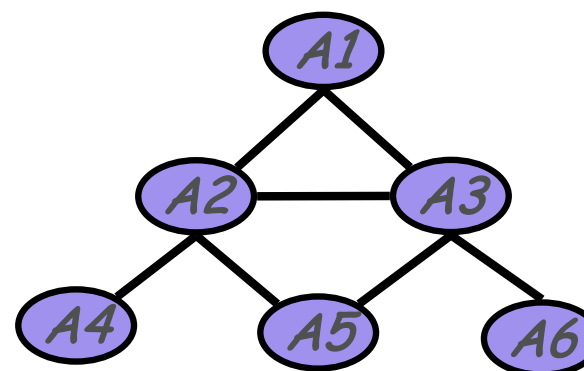
$$\begin{aligned} &P(A_1) \\ &P(A_2|A_1) \\ &P(A_3|A_1) \\ &P(A_4|A_2) \\ &P(A_5|A_2, A_3) \\ &P(A_6|A_3) \end{aligned}$$

# Moralizing



Bayesian Network

Moralizing  
→



Domain Graph  
(Moral Graph)

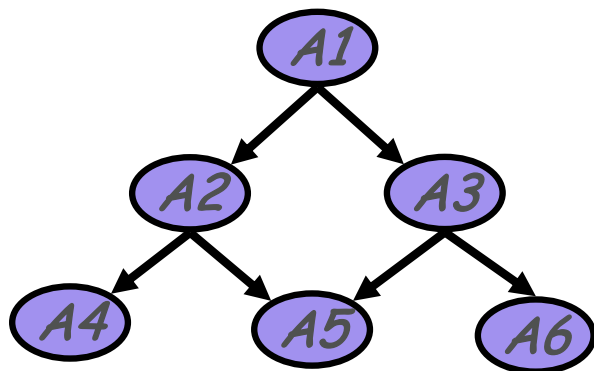
CPDs  
↙

$P(A_1)$   
 $P(A_2|A_1)$   
 $P(A_3|A_1)$   
 $P(A_4|A_2)$   
 $P(A_5|A_2, A_3)$   
 $P(A_6|A_3)$

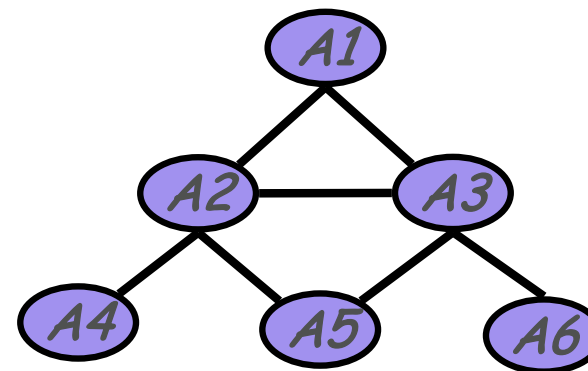
→  
CPD  
=  
potential

$dom(\phi_1) = \{A_1\}$   
 $dom(\phi_2) = \{A_2, A_1\}$   
 $dom(\phi_3) = \{A_3, A_1\}$   
 $dom(\phi_4) = \{A_4, A_2\}$   
 $dom(\phi_5) = \{A_5, A_2, A_3\}$   
 $dom(\phi_6) = \{A_6, A_3\}$

# Moralizing



Moralizing



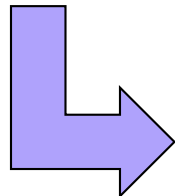
Bayesian Network

Domain Graph  
(Moral Graph)

Potentials induce  
undirected edges



CPDs

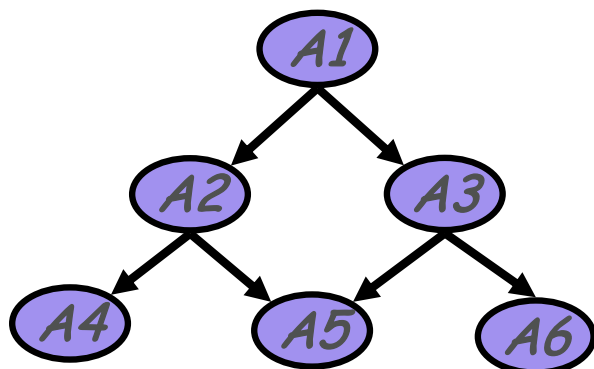


$P(A_1)$   
 $P(A_2|A_1)$   
 $P(A_3|A_1)$   
 $P(A_4|A_2)$   
 $P(A_5|A_2, A_3)$   
 $P(A_6|A_3)$

CPD  
=  
potential

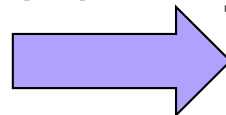
$dom(\phi_1) = \{A_1\}$   
 $dom(\phi_2) = \{A_2, A_1\}$   
 $dom(\phi_3) = \{A_3, A_1\}$   
 $dom(\phi_4) = \{A_4, A_2\}$   
 $dom(\phi_5) = \{A_5, A_2, A_3\}$   
 $dom(\phi_6) = \{A_6, A_3\}$

# Moralizing

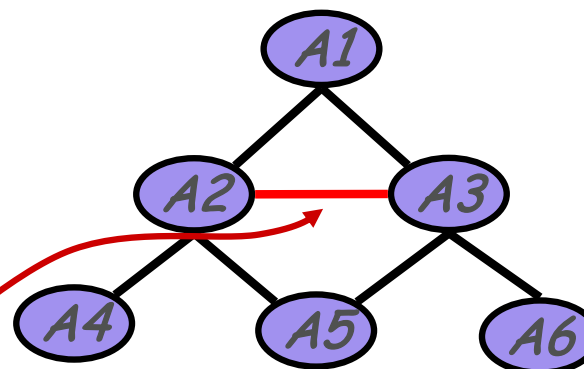


Bayesian Network

Moralizing

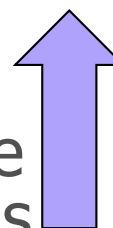


moral link

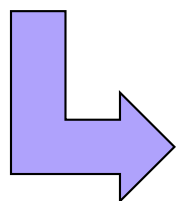


Domain Graph  
(Moral Graph)


Potentials induce  
undirected edges



CPDs



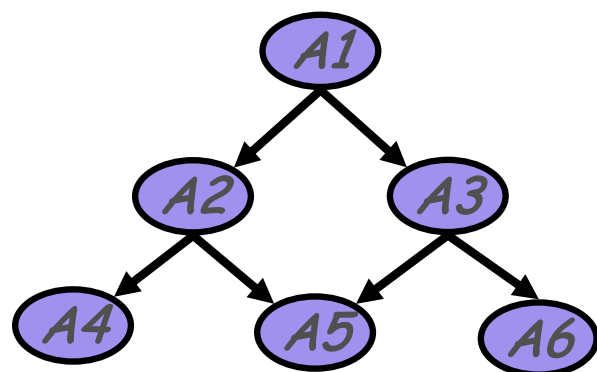
$P(A_1)$   
 $P(A_2|A_1)$   
 $P(A_3|A_1)$   
 $P(A_4|A_2)$   
 $P(A_5|A_2, A_3)$   
 $P(A_6|A_3)$

  
 CPD  
 =  
 potential

$dom(\phi_1) = \{A_1\}$   
 $dom(\phi_2) = \{A_2, A_1\}$   
 $dom(\phi_3) = \{A_3, A_1\}$   
 $dom(\phi_4) = \{A_4, A_2\}$   
 $dom(\phi_5) = \{A_5, A_2, A_3\}$   
 $dom(\phi_6) = \{A_6, A_3\}$

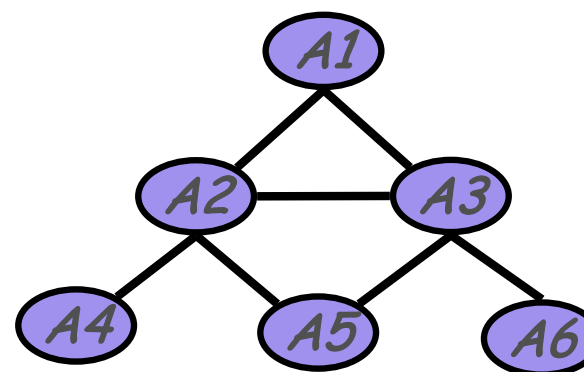
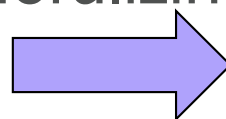


# VE on the Moral Graph: Elimination Sequence



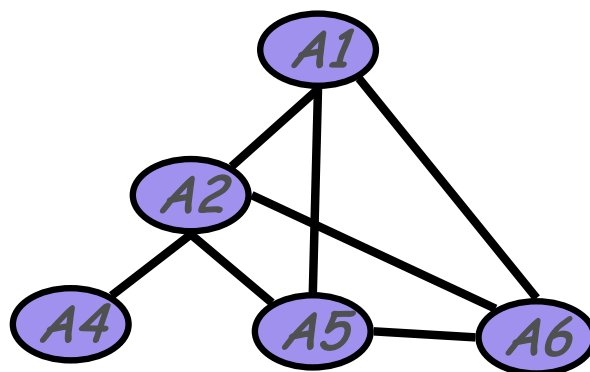
Bayesian Network

Moralizing

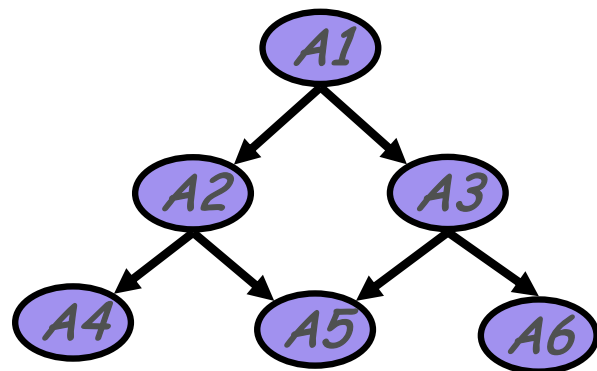


Moral Graph

Eliminating A3

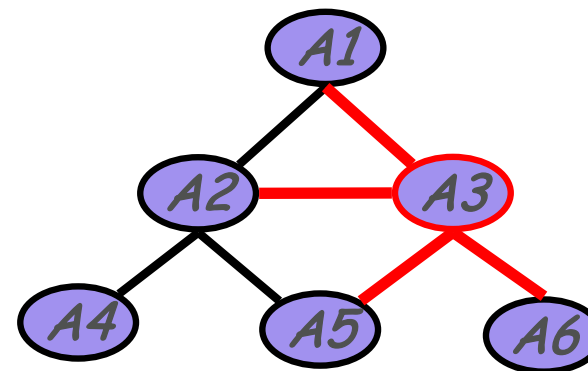
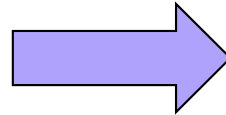


# Elimination Sequence



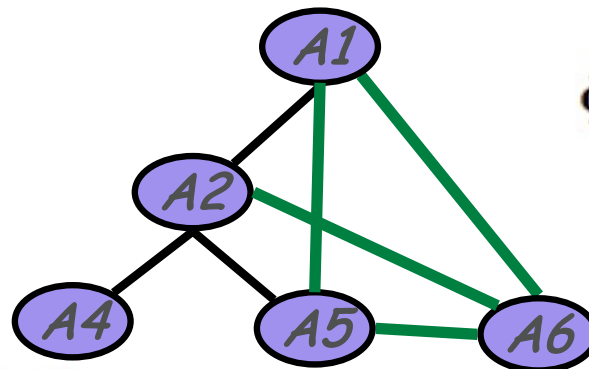
Bayesian Network

Moralizing



Moral Graph

Eliminating A3



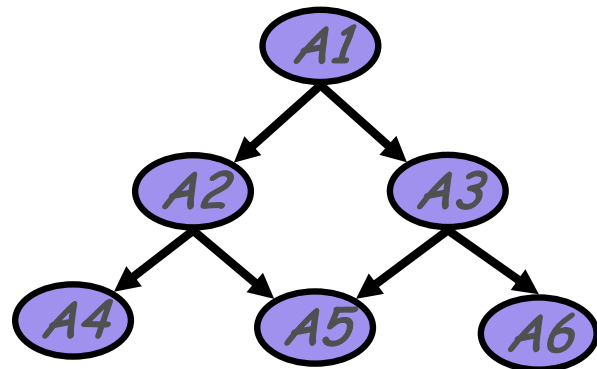
$$\begin{aligned}\phi_1 &= P(A_1) \\ \phi_2 &= P(A_2|A_1) \\ \phi_3 &= P(A_3|A_1) \\ \phi_4 &= P(A_4|A_2) \\ \phi_5 &= P(A_5|A_2, A_3) \\ \phi_6 &= P(A_6|A_3)\end{aligned}$$

$$\Phi^{-A_3} = \sum_{A_3} \Phi_3 \cdot \Phi_5 \cdot \Phi_6$$

Domain: A1, A2, A5, A6

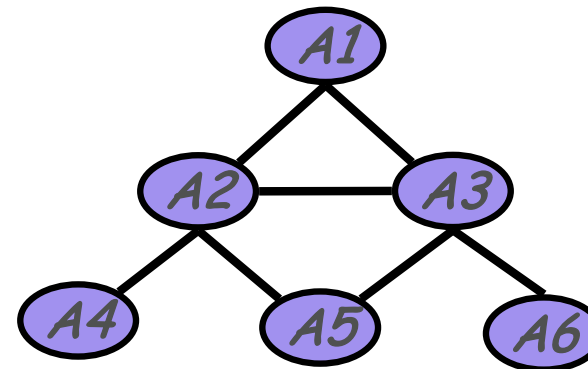
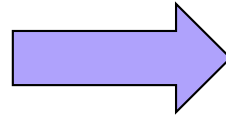
# Elimination Sequence

**GOAL: Elimination sequence that does not introduce fill-ins**



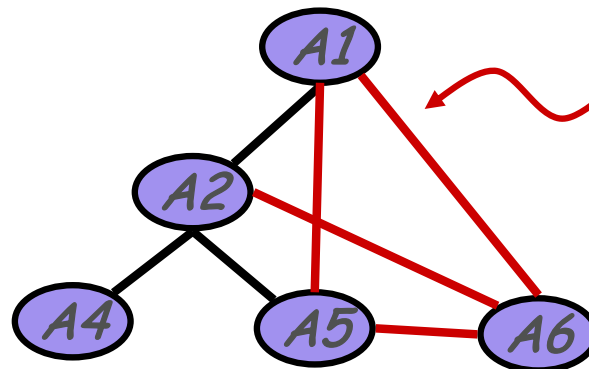
Bayesian Network

Moralizing



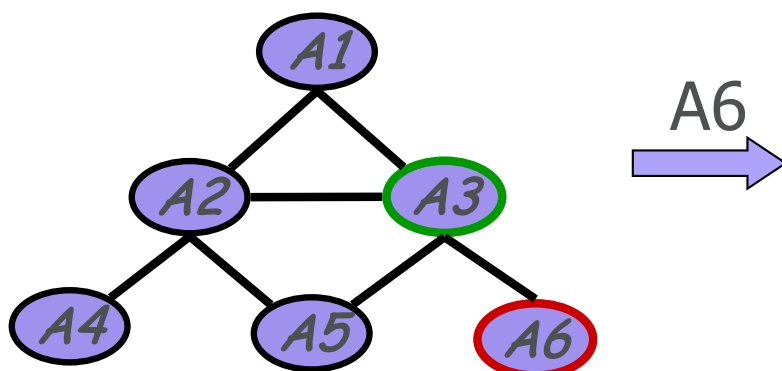
Moral Graph

Eliminating A3

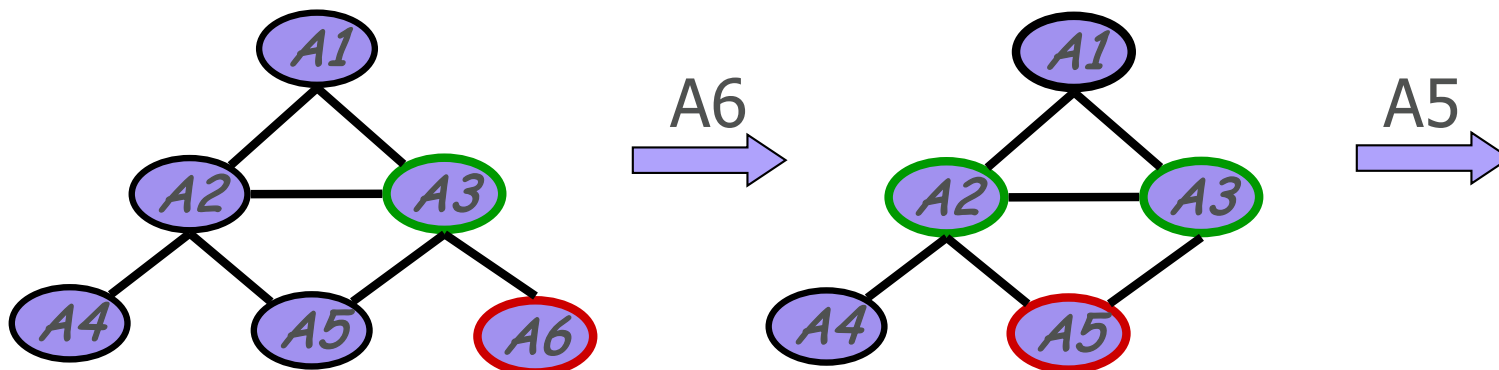


**fill-ins:** work with a potential over a domain that was not present originally

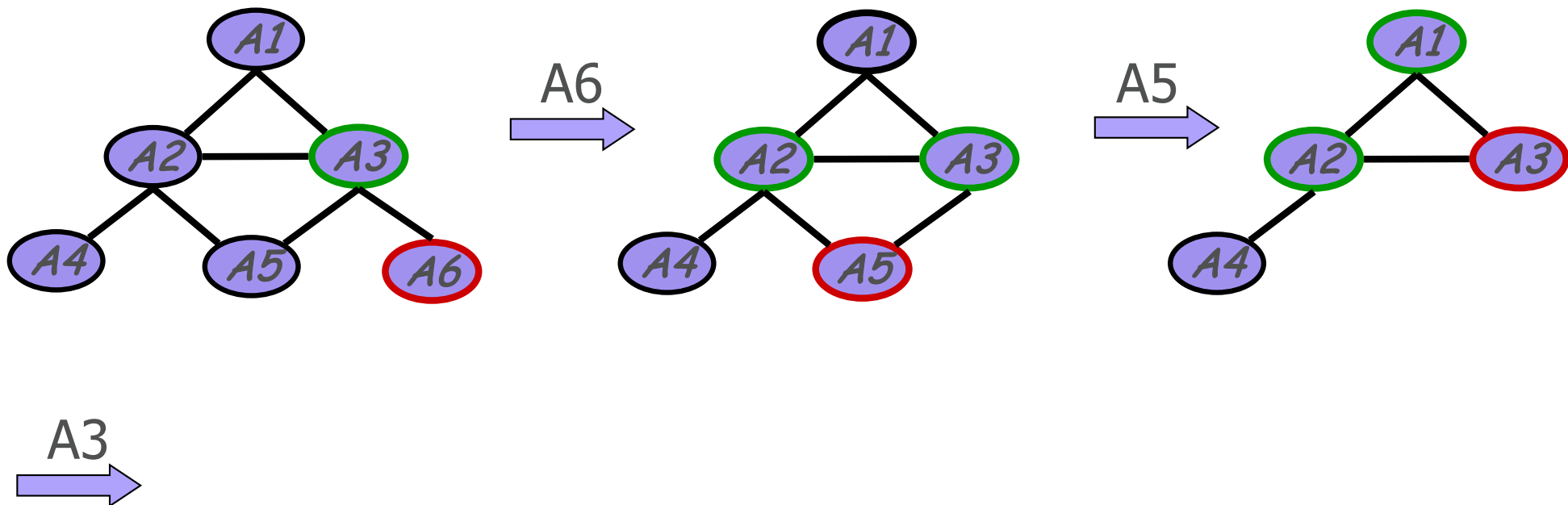
# Perfect Elimination Sequence ...



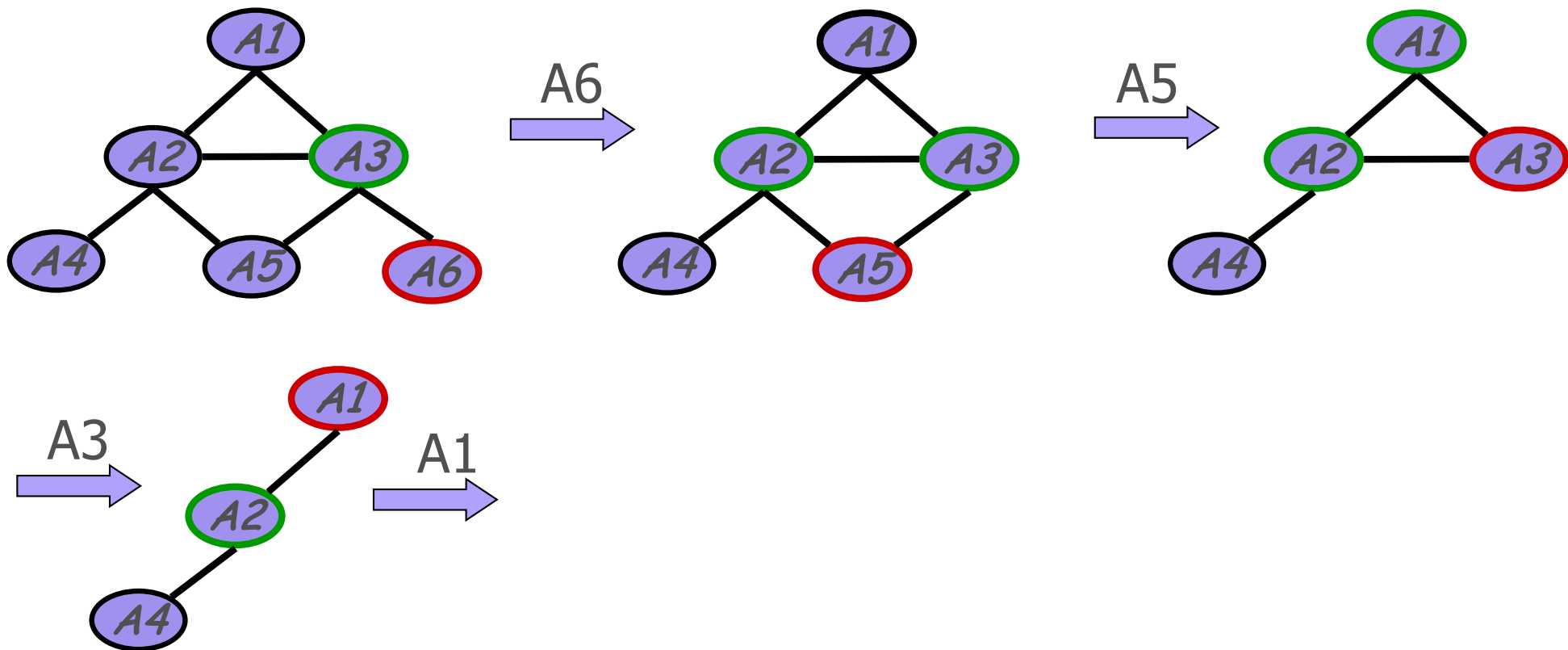
# Perfect Elimination Sequence ...



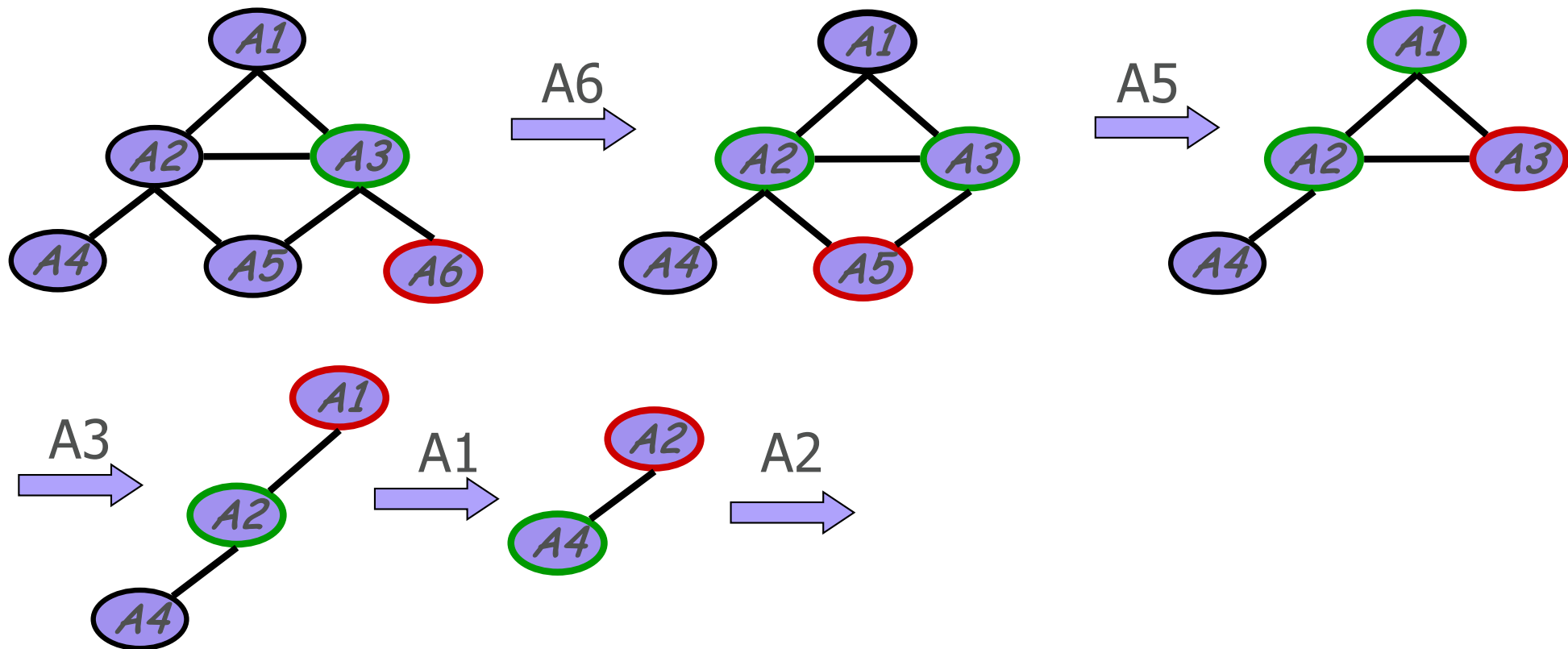
# Perfect Elimination Sequence ...



# Perfect Elimination Sequence ...

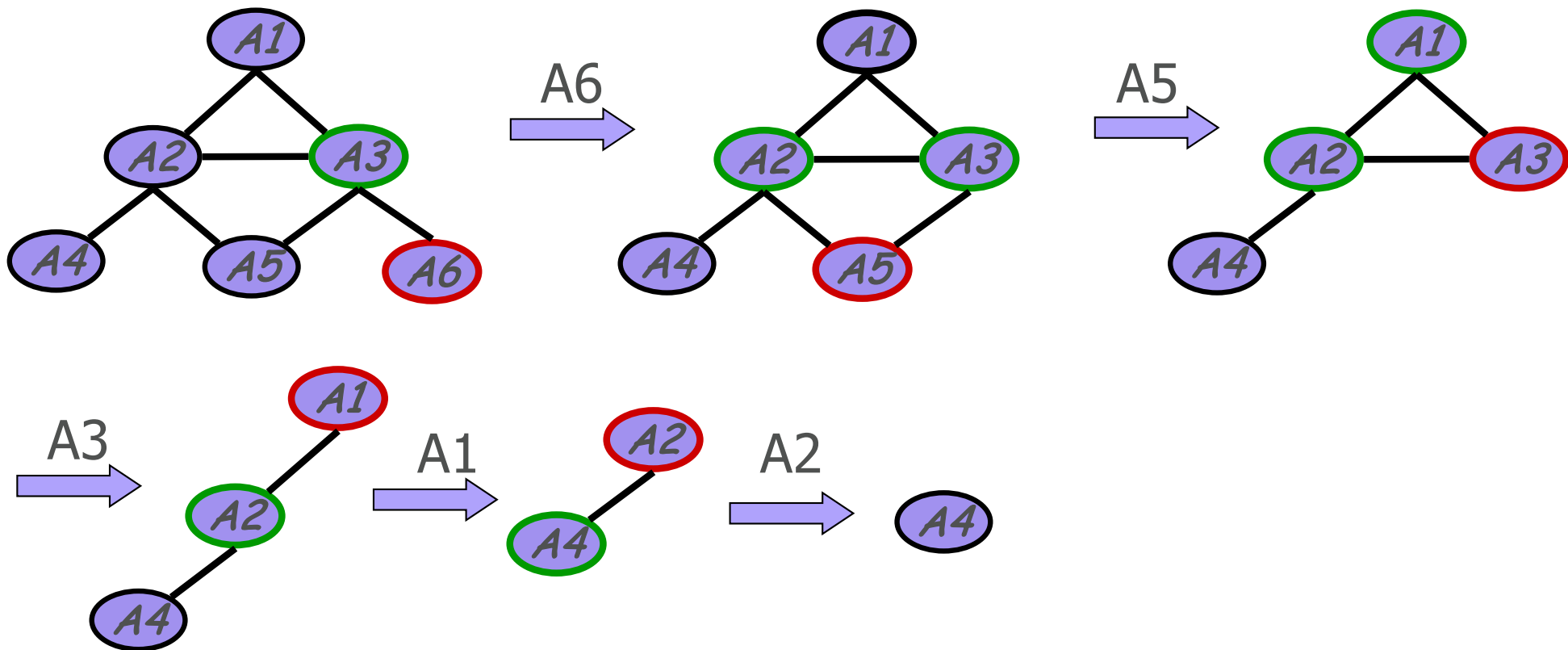


# Perfect Elimination Sequence ...



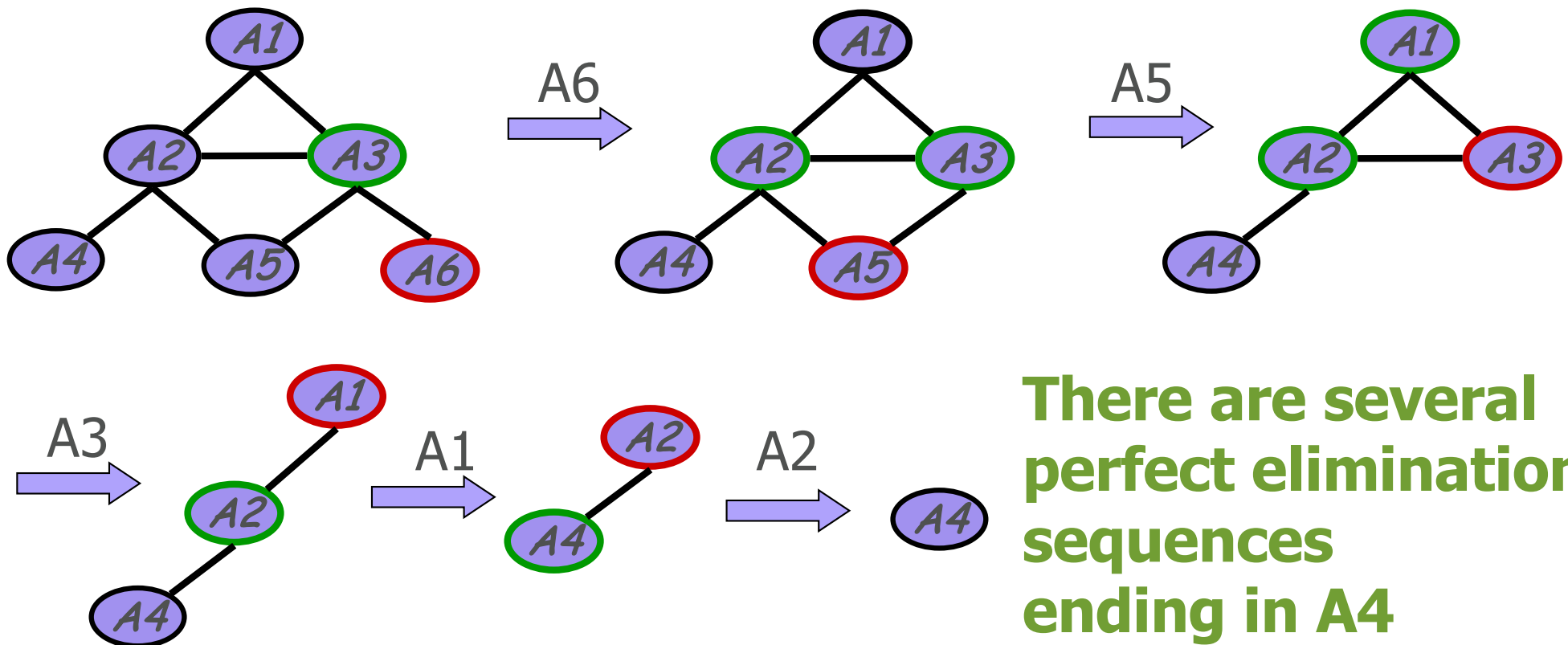


# Perfect Elimination Sequence ...



# Perfect Elimination Sequence ...

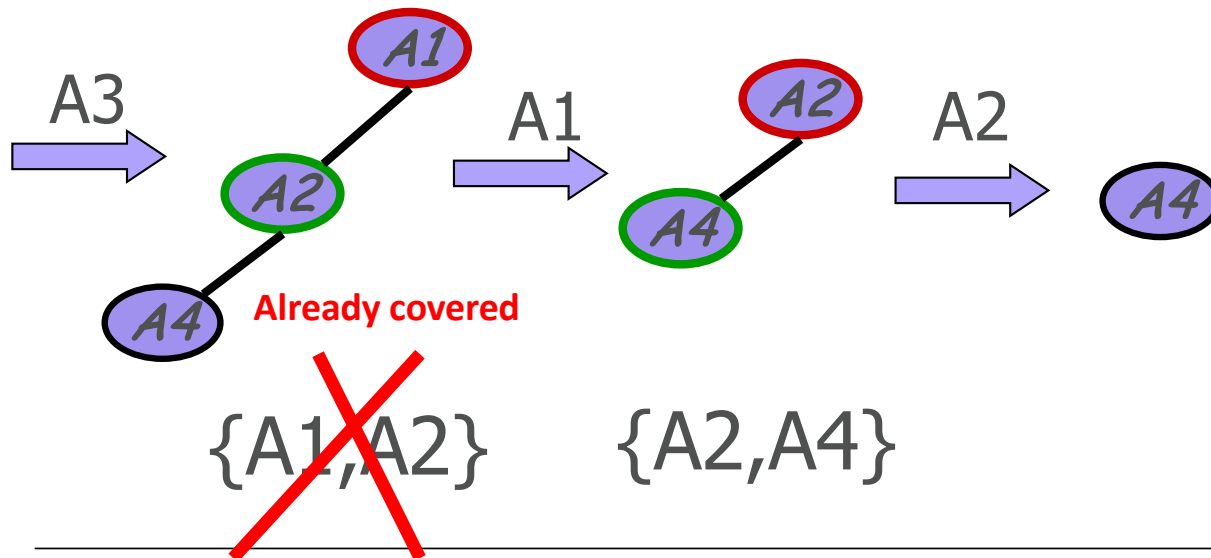
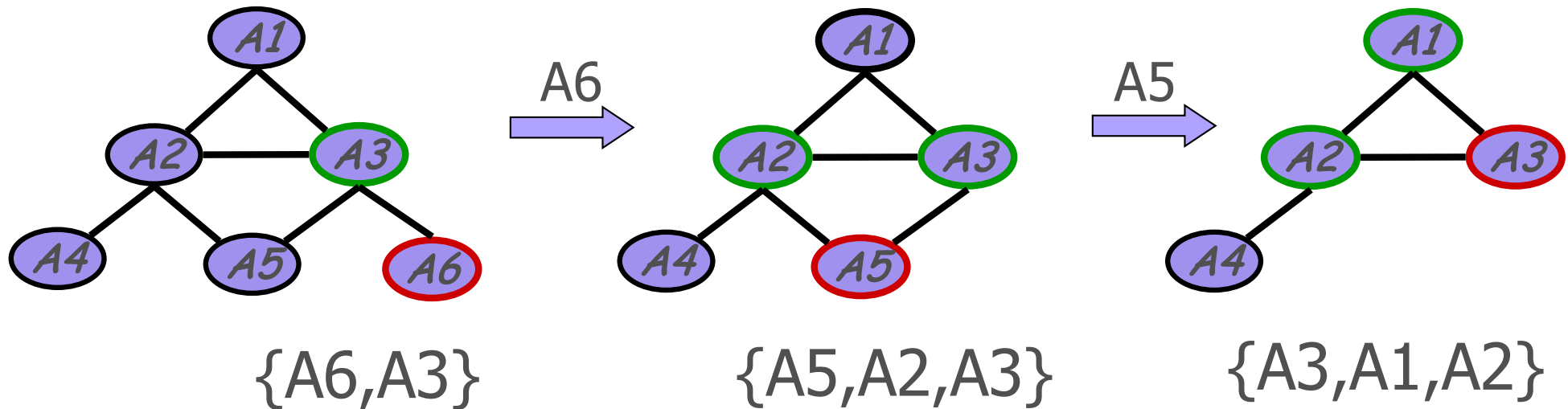
... do not introduce fill-ins



# Complexity of VE = Complexity of Elimination Sequence

- Characterized by the set of domains
- Set of domains of potentials produced during the elimination (**potentials** that are subsets of other potentials are removed).
- A6,A5,A3,A1,A2,A4:  $\{\{A6,A3\},\{A2,A3,A5\},\{A1,A2,A3\},\{A2,A4\}\}$

# Complexity of Elimination Sequence



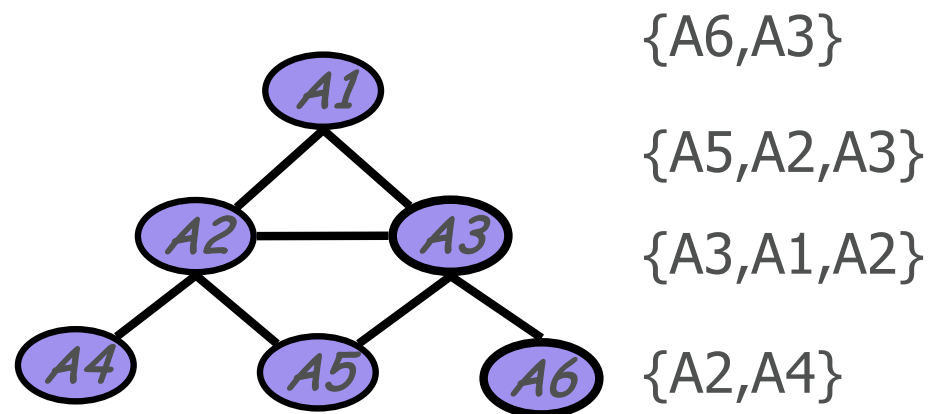
# Complexity of Elimination Sequence

- All perfect elimination sequences produce the same domain set, namely the set of cliques of the domain
- Any **perfect elimination sequence** ending with  $A$  is **optimal** with respect to **computing  $P(A)$**

# Induced Graph

- The induced graph for an elimination order  $O$  has an edge  $X_i - X_j$  if  $X_i$  and  $X_j$  appear together in a factor “generated” by VE for elimination order  $O$  on factors/potentials  $F$  (moral graph is a subgraph)

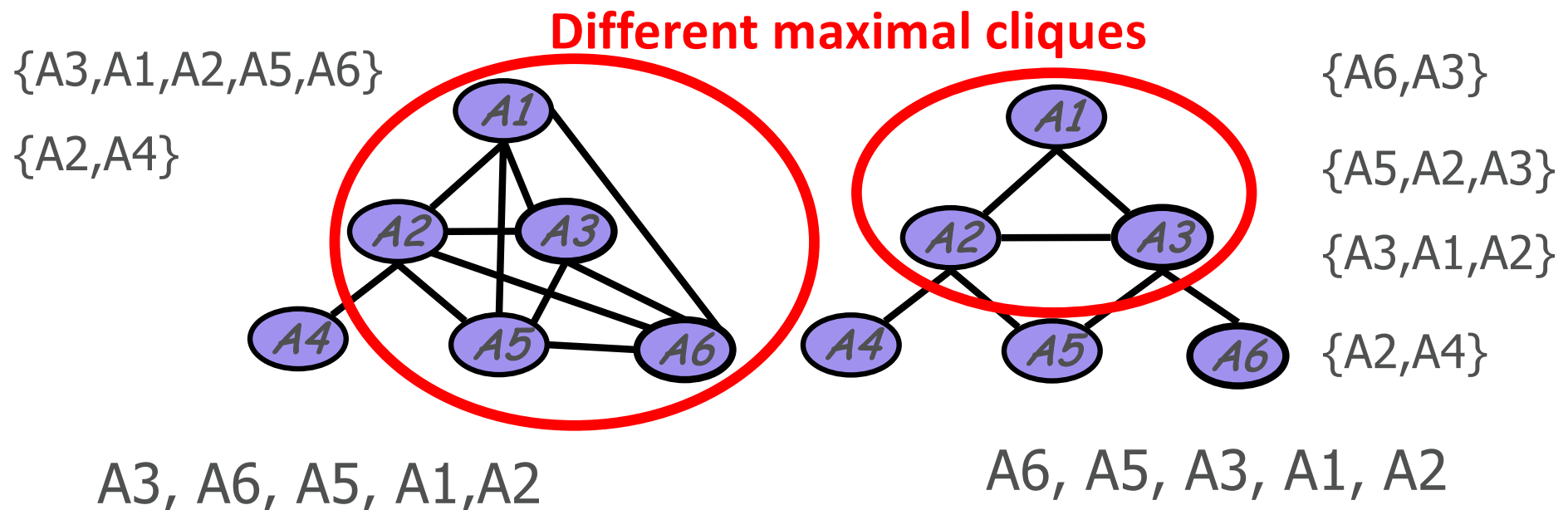
maximal cliques that  
cover the induced  
graph



**Elimination Order:**  
 $A6, A5, A3, A1, A2$

# Induced Graph

- The induced graph for an elimination order  $O$  has an edge  $X_i - X_j$  if  $X_i$  and  $X_j$  appear together in a factor “generated” by VE for elimination order  $O$  on factors/potentials  $F$  (moral graph is a subgraph)



# Complexity of VE = Complexity of Elimination Sequence

- Main property of induced Graph:
  - Every maximal clique in the induced graph corresponds to an intermediate factor in the VE computations
  - Every factor stored during the VE process is a subset of some maximal clique in the graph
- These facts are true for any variable elimination ordering on any network

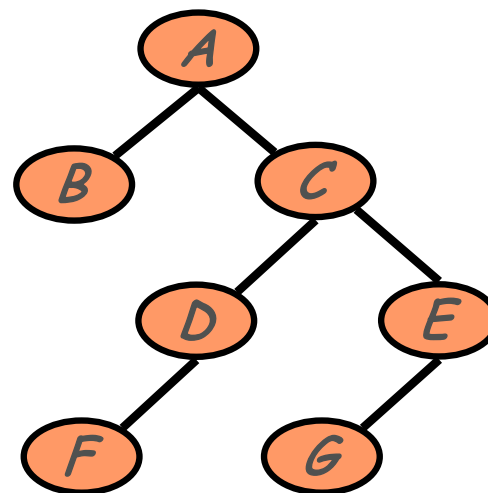
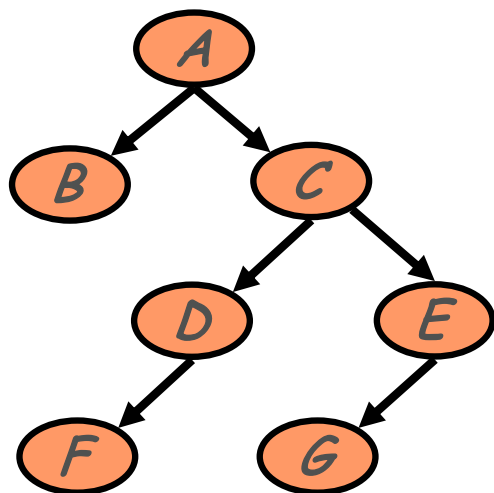


# Induced Width (Treewidth)

- The size of the largest clique in the induced graph is thus an indicator for the complexity of variable elimination
- This quantity (minus one) is called the **induced width** (or **treewidth**) of a graph according to the specified ordering
- Finding a good ordering for a graph is equivalent to finding the minimal induced width of the graph
- Finding an ordering with minimal induced-width is NP-complete

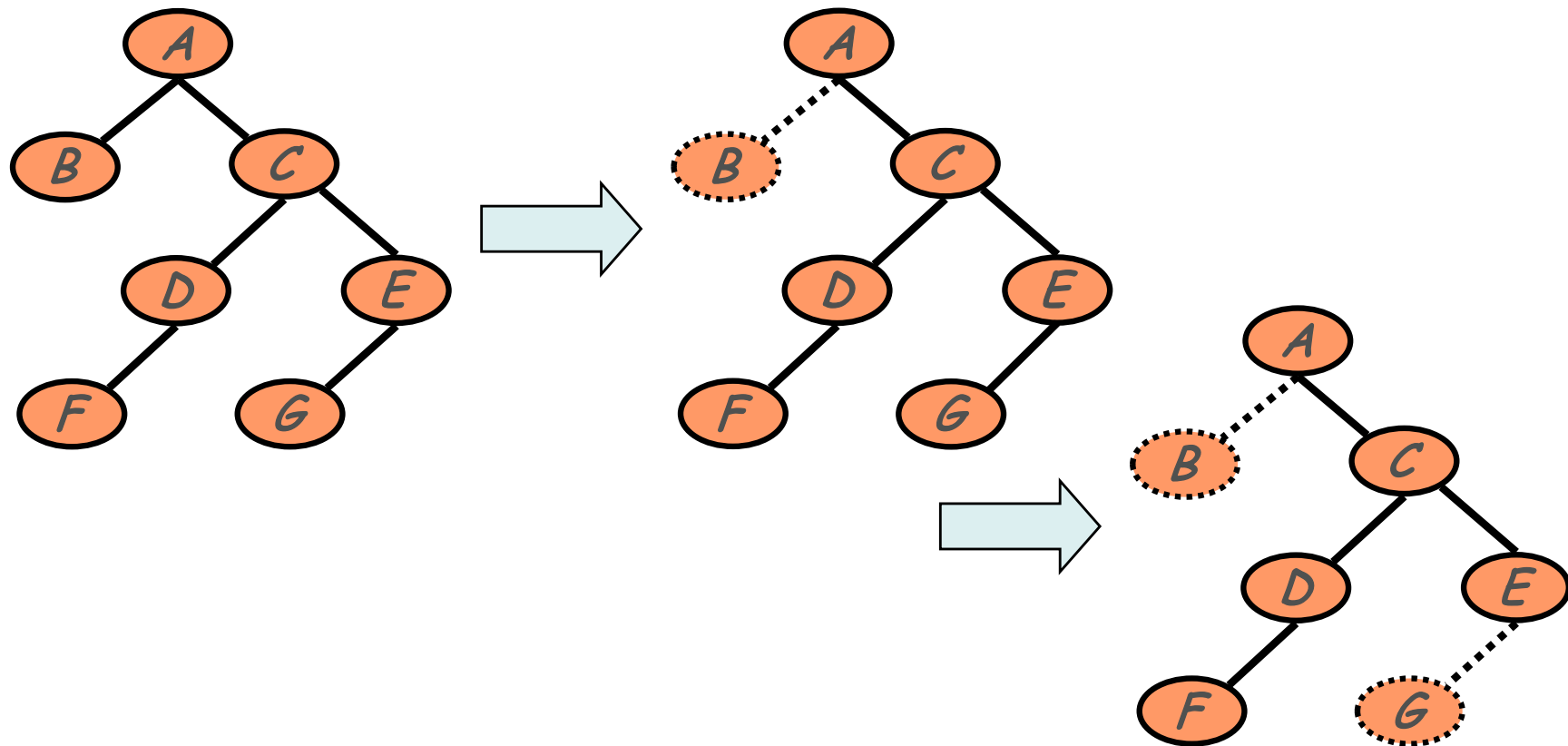
# Consequence: Elimination on Trees

- Suppose we have a **tree**, i.e., a network where each variable has at most one parent. Then:
- All the factors involve at most two variables
- Thus, the moralized graph is also a tree



# Elimination on Trees

- We can maintain the tree structure by eliminating extreme variables in the tree



# Elimination on Trees

- Formally, for any tree, there is an elimination ordering with **treewidth = 1**

## Theorem

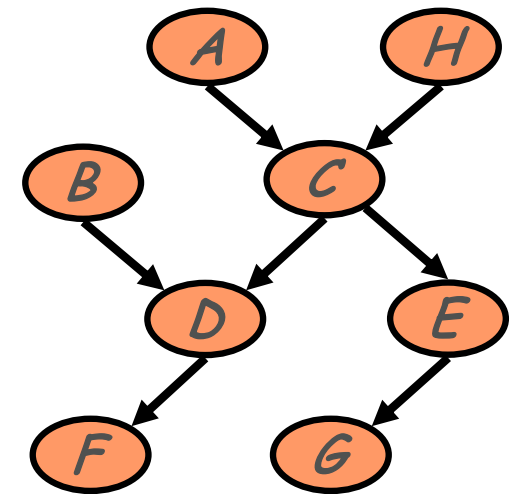
- Inference on trees is linear in number of variables

# Polytrees

- A polytree is a network where there is at most one path from one variable to another

## Theorem:

- Inference in a polytree is linear in the representation size of the network
  - This assumes tabular CPT representation
- Can you see how the argument would work? Maybe this will be a HW



# General Networks

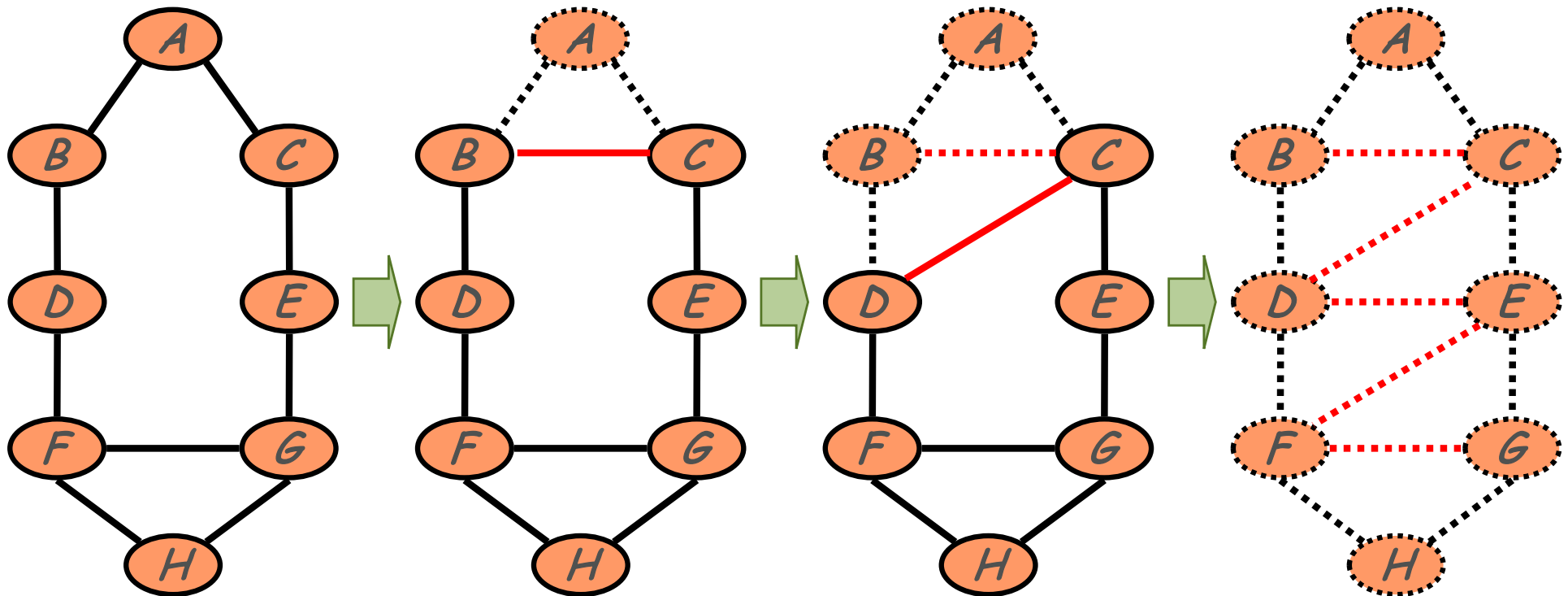
What do we do when the network is not a polytree?

- If network has a cycle, the treewidth for any ordering is greater than 1

## Example

- Eliminating A, B, C, D, E,....
- Resulting graph has treewidth 2

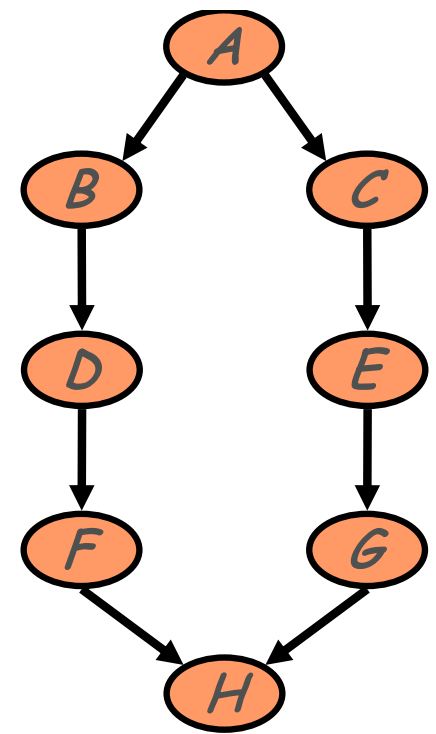
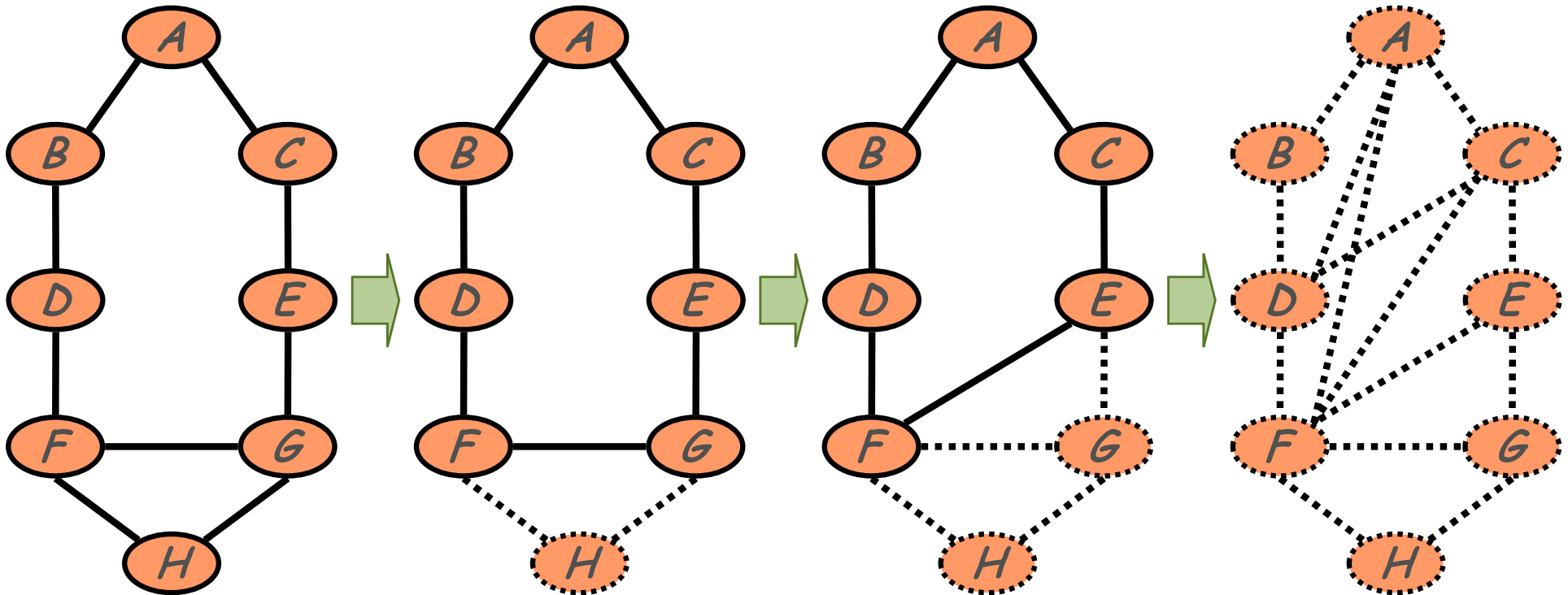
Moral Graph



## Another Example

- Eliminating H, G, E, C, F, D, B, A

Moral Graph





# General Networks

- From graph theory:
  - **Theorem: Finding an ordering that minimizes the treewidth is NP-Hard**

However,

- There are reasonable heuristics for finding “relatively” good ordering
- There are provable approximations to the best treewidth
- If the graph has a small treewidth, there are algorithms that find it in polynomial time

# Summary Complexity Results

- Probabilistic inference

- general graphs:
- poly-trees and low tree-width:

#P-complete

easy

- Approximate probabilistic inference

- Absolute error:  $|\hat{P} - P| \leq \varepsilon \dots$  NP-hard  $\forall \varepsilon < 0.5$
- Relative error:  $\frac{|\hat{P} - P|}{P} \leq \varepsilon \dots$  NP-hard  $\forall \varepsilon > 0$

- Most probable explanation (MPE)

- general graphs:
- poly-trees and low tree-width:

NP-complete

easy

- Maximum a posteriori (MAP)

- general graphs:
- poly-trees and low tree-width:

$\text{NP}^{\text{PP}}$ -complete

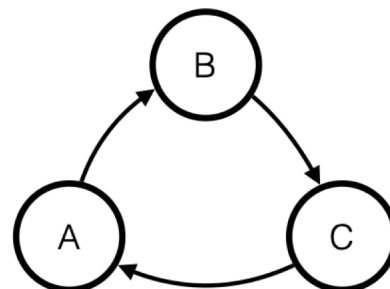
NP-hard

# What you need to know about inference thus far

- **Variable elimination algorithm**
  - Eliminate a variable:
    - Combine factors that include this var into single factor
    - Marginalize var from new factor
  - **Efficient algorithm (“only” exponential in induced-width, not number of variables)**
    - If you hear: “Exact inference only efficient in tree graphical models”
    - You say: “No!!! Any graph with low induced width”
    - And then you say: “And even some with very large induced-width” (with context-specific independence)
- **Elimination order is important!**
  - **NP-complete problem**
  - Many good heuristics

# Acyclicity of Bayes Nets

invalid Bayes net

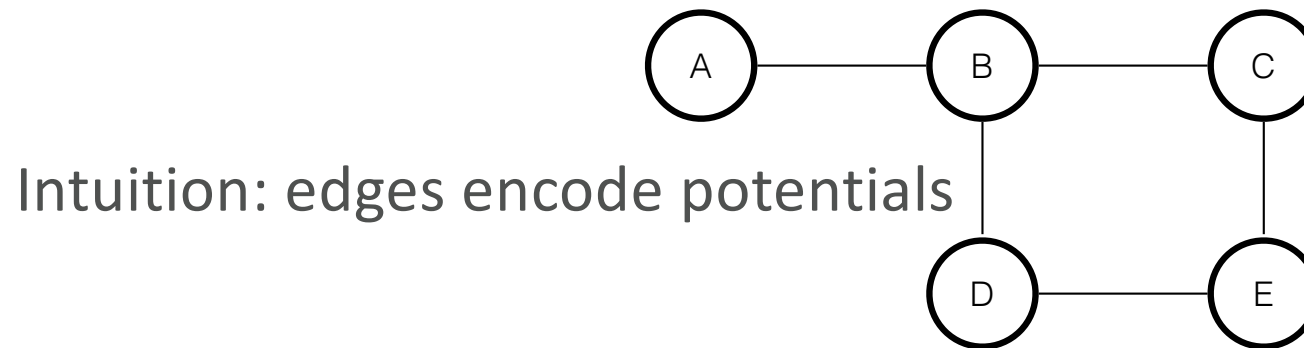


Only “makes sense” if  $P(A) = P(B) = P(C) = 1$

All meaningful Bayes nets are directed, acyclic graphs (DAGs)

We need a different form

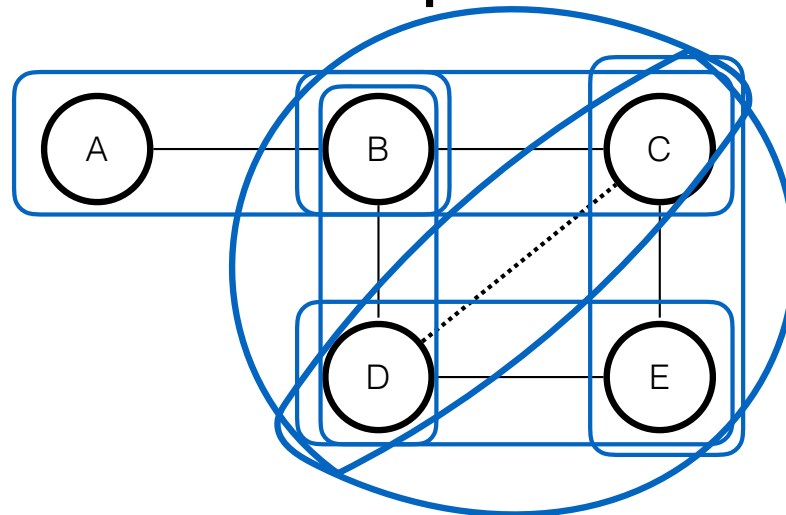
# Undirected Graphical Models



$$P(A, B, C, D, E) \propto \phi(A, B)\phi(B, C)\phi(B, D)\phi(C, E)\phi(D, E)$$

$$P(X) = \frac{1}{Z} \prod_{c \in \text{cliques}(G)} \phi_c(x_c) \quad \text{potential functions}$$

# Undirected Graphical Models



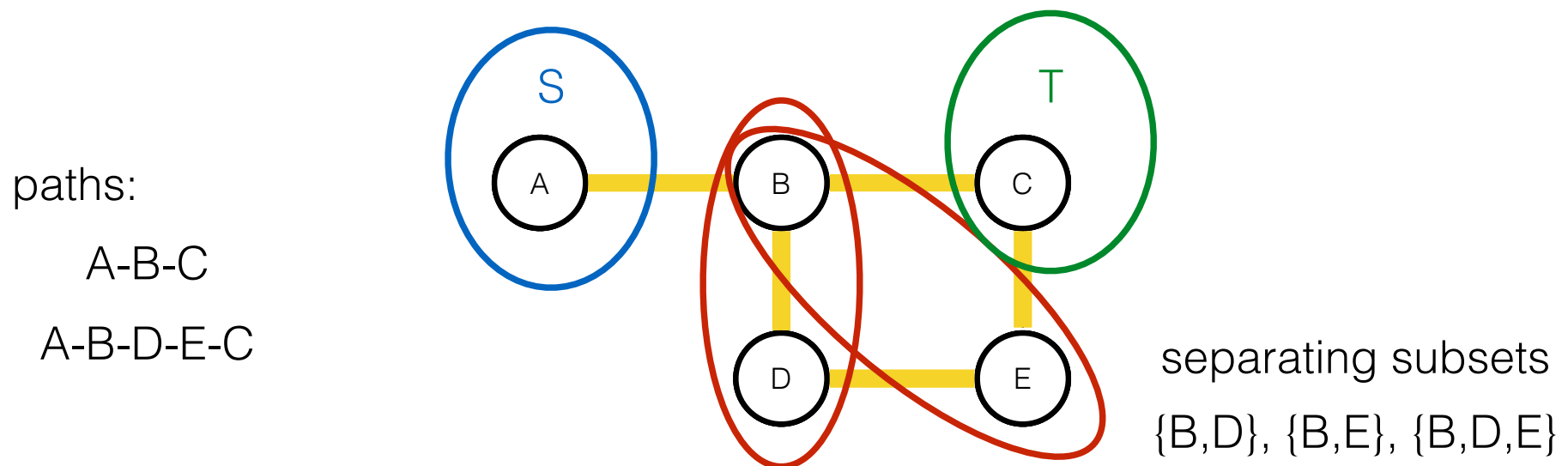
$$P(A, B, C, D, E) \propto \cancel{\phi(A, B)\phi(B, C)\phi(B, D)\phi(C, E)\phi(D, E)} \\ \phi(A, B)\phi(B, C, D)\phi(C, D, E)$$

$$P(X) = \frac{1}{Z} \prod_{c \in \text{cliques}(G)} \phi_c(x_c) \quad \text{potential functions}$$

Markov Random Fields (MRFs) have  
independency properties that obey the Markov property

# Markov Random Fields

- Any two subsets  $S$  and  $T$  of variables are conditionally independent given a **separating subset**
- All paths between  $S$  and  $T$  must travel through the separating subset



## Other (implied) independencies

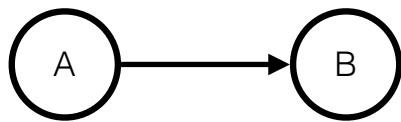
# Independence Corollaries

- Any two non-adjacent variables are conditionally independent given all other variables
- Any variable is conditionally independent of the other variables given its neighbors
- Markov blanket

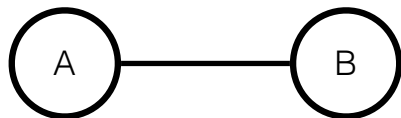


So how to BNs and MRFs relate?

# Bayesian Networks as MRFs



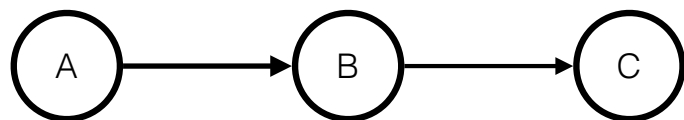
$$p(A, B) = p(A)p(B|A)$$



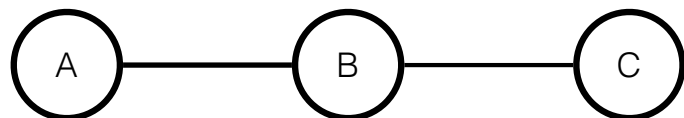
$$p(A, B) \propto \phi(A, B)$$

converting a single edge to a pairwise clique potential is easy

# Bayesian Networks as MRFs



$$p(A, B, C) = p(A)p(B|A)P(C|B)$$



$$p(A, B, C) \propto \phi(A, B)\phi(B, C)$$

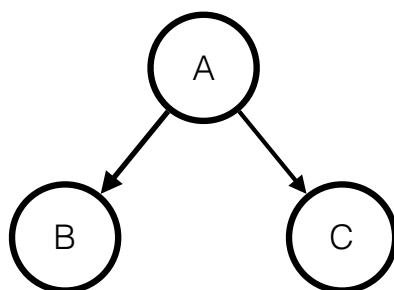
$$\phi(A, B) \leftarrow P(A)P(B|A)$$

$$\phi(B, C) \leftarrow P(C|B)$$

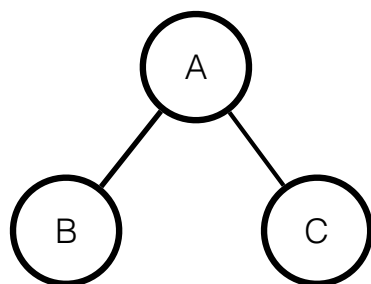
chains are easy too

parameterization is not unique

# Bayesian Networks as MRFs



$$p(A, B, C) = p(A)P(B|A)P(C|A)$$



$$p(A, B, C) \propto \phi(A, B), \phi(A, C)$$

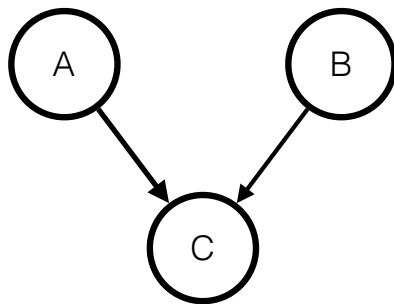
$$\phi(A, B) \leftarrow P(A)P(B|A)$$

$$\phi(A, C) \leftarrow P(C|A)$$

shared parents also easy

Now the v-structure! Can we also capture the BN in the MRF?

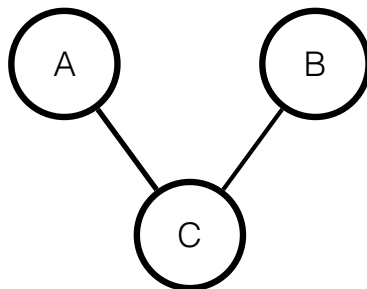
# Bayesian Networks as MRFs



$$p(A, B, C) = p(A)p(B)p(C|A, B)$$

A and B are **dependent** given C

What if we observe C?



$$p(A, B, C) \propto \phi(A, C)\phi(B, C)$$

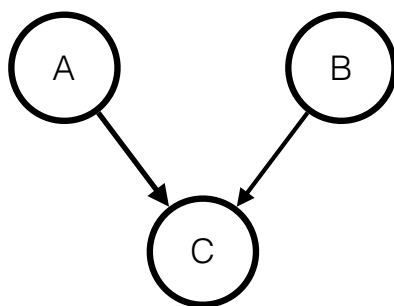
A and B are **independent** given C

can't be correct

shared child

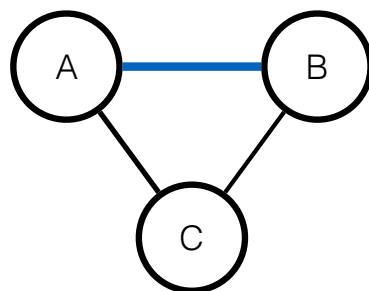
The solution!

# Moralizing Parents



$$p(A, B, C) = p(A)p(B)p(C|A, B)$$

A and B are **dependent** given C



$$p(A, B, C) \propto \phi(A, C)\phi(B, C) \phi(A, B)$$

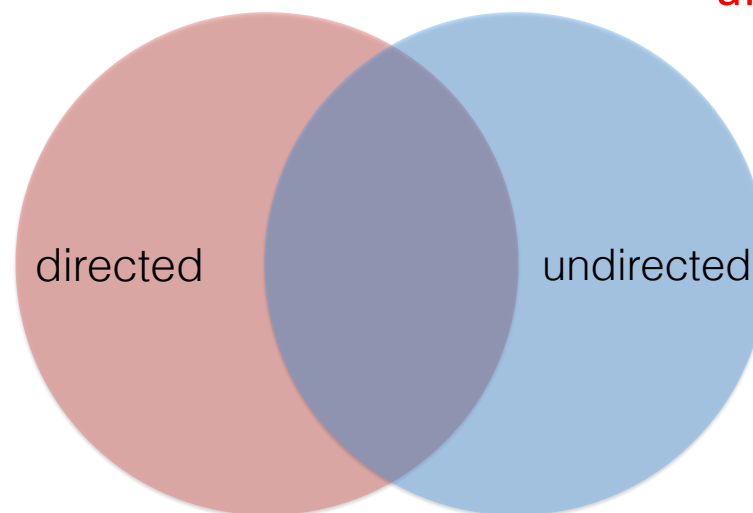
A and B are **independent** given C

shared child

Explains the moral link we talked about already

# Converting Bayes Nets to MRFs

- Moralize all co-parents
- Lose marginal independence of parents (v-structure: A and B are now almost always dependent when not observing c))



## Back to inference

### What if we have to run multiple inferences?

- Multiple inference, e.g.,

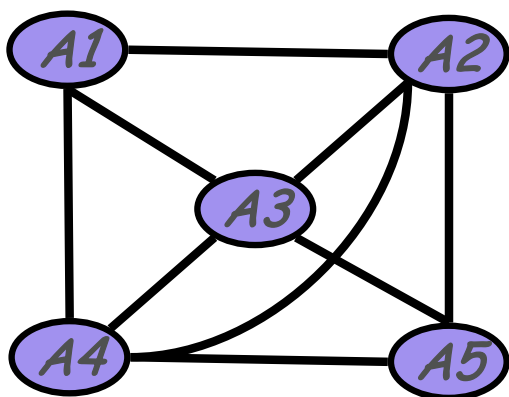
$$P(X_i, e) \qquad P(X_i|e) = \frac{P(X_i, e)}{P(e)}$$

- For each  $i$  do variable elimination?

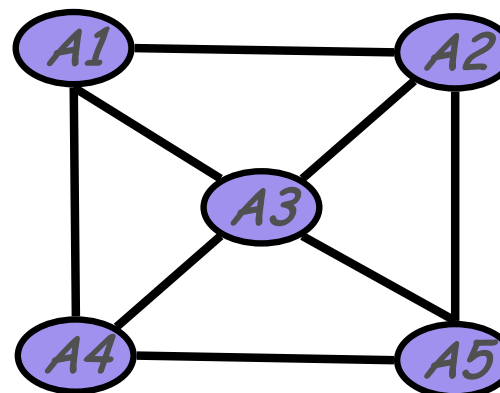
**No, instead reuse information  
resp. computations**

# Triangulated Graphs and Join Trees

- An undirected graph with **perfect elimination sequence** (no fill-ins) is called a *triangulated* graph



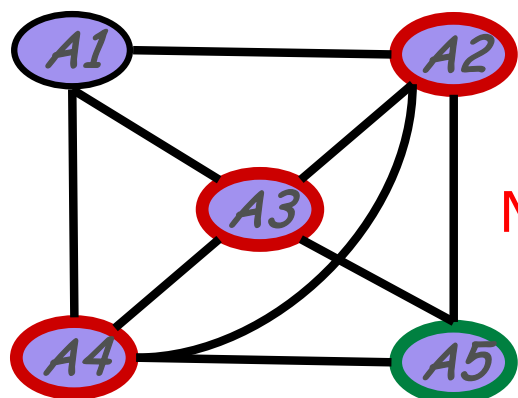
triangulated



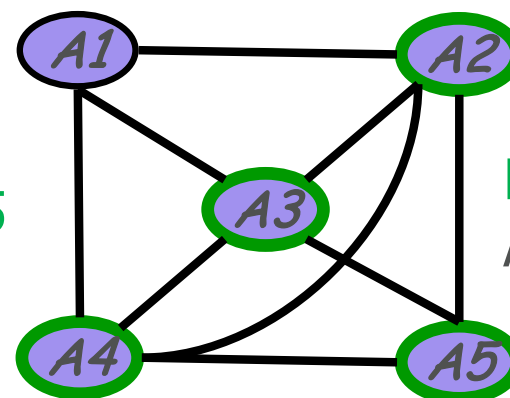
nontriangulated



# Triangulated Graphs and Join Trees

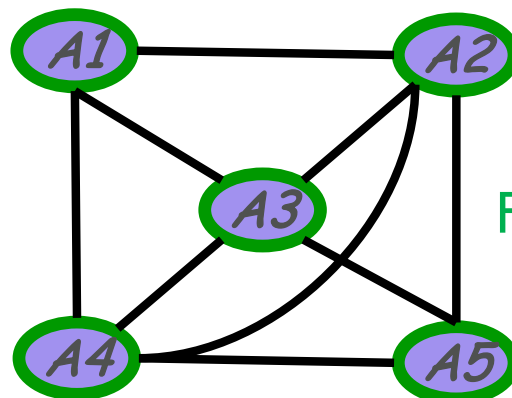


Neighbours of A5



Family of A5, i.e.,  
A5 is simplicial

- Complete neighbour set = all neighbours are pairwise linked = simplicial node
- X is simplicial iff family of X is a clique



Family of A3, i.e. A3 is **not** simplicial

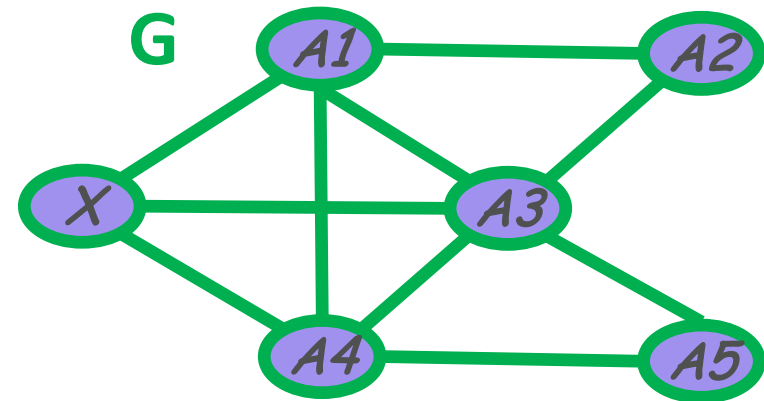
# Triangulated Graphs and Join Trees

Let  $G$  be a **triangulated** graph,  
and let  $X$  be a **simplicial** node.

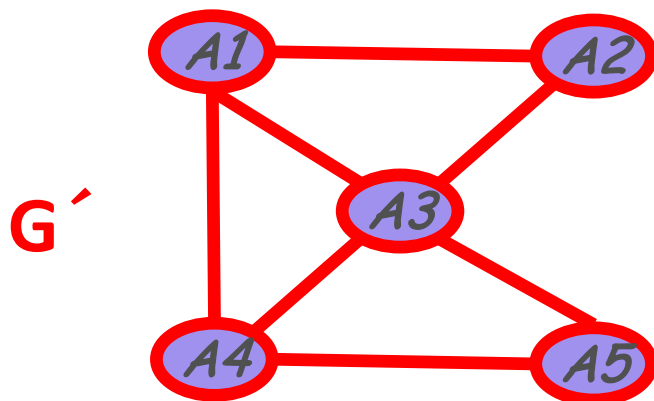
Let  $G'$  be the graph resulting  
from **eliminating  $X$**

(including its edges) from  $G$ .

**Then  $G'$  is a triangulated graph**



Eliminating  $X$



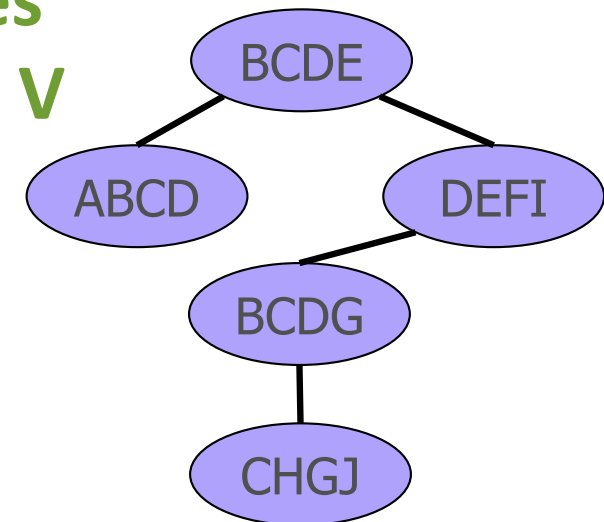
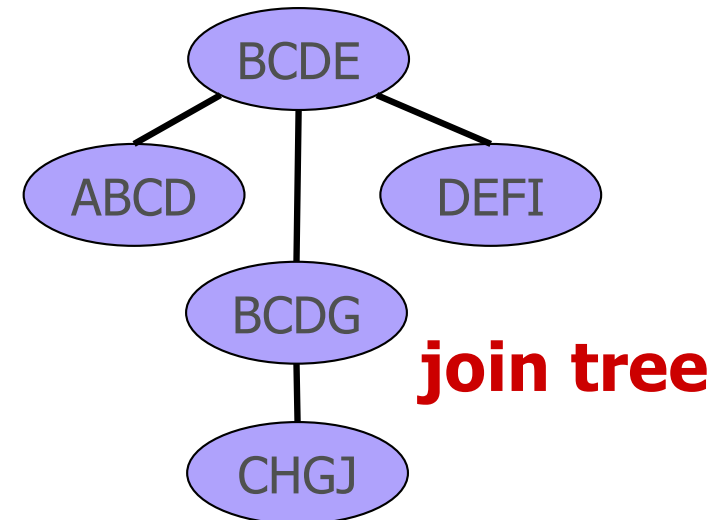
# Triangulated Graphs and Join Trees

- A triangulated graph with at least two nodes has at least two simplicial nodes
- In a triangulated graph, each variable  $A$  has a perfect elimination sequence ending with  $A$
- Not all domain graphs are triangulated: An undirected graph is triangulated iff **all nodes can be eliminated by successively eliminating a simplicial node**

# Join Trees

Let  $G$  be the set of cliques from an undirected graph, and let the cliques of  $G$  be organized in a tree

- **T is a join tree** if for any pair of nodes  $V, W$  all nodes on the path between  $V$  and  $W$  contain the intersection  $V \cap W$
- This is called „Running Intersection Property“



**not a join tree**

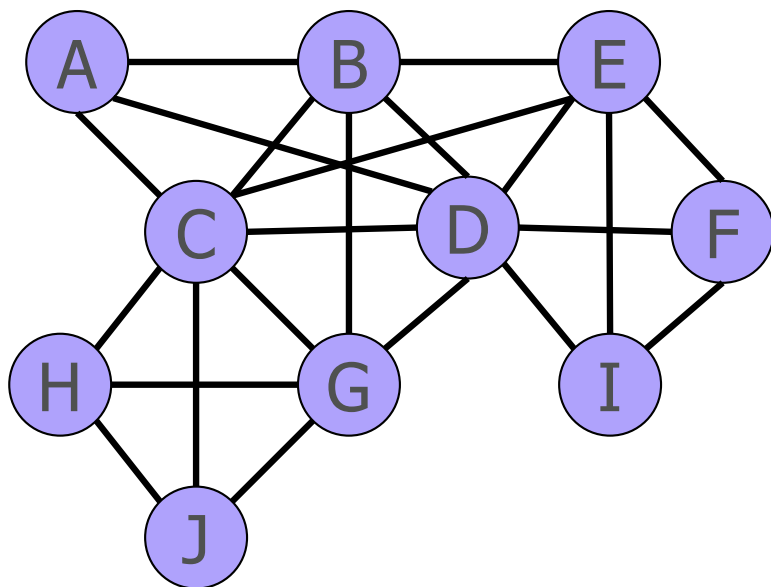
# Join Trees $\Leftrightarrow$ Triangulated

It can be shown that

- If the cliques of an undirected graph  $G$  can be organized into a join tree, then  $G$  is triangulated
- If the undirected graph is triangulated, then the cliques of  $G$  can be organized into a join tree

# Triangulated, undirected Graph

## -> Join trees

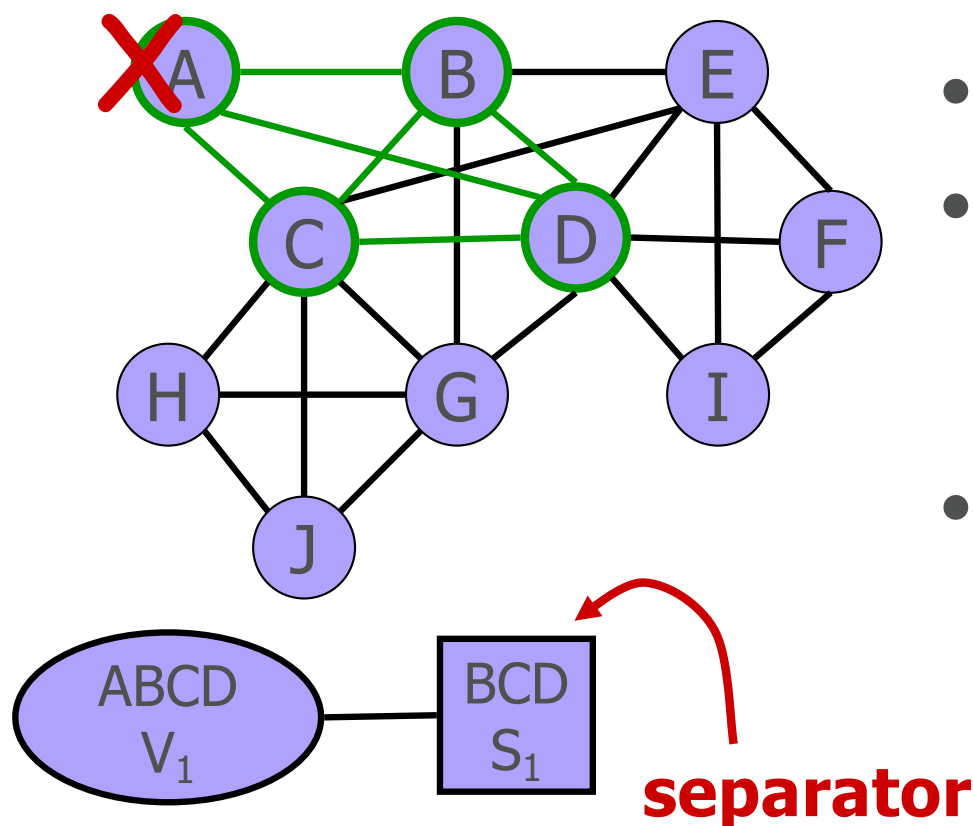


- Simplicial node  $X$
- Family of  $X$  is a clique
- Eliminate nodes from family of  $X$  which have only neighbours in the family of  $X$
- Give family of  $X$  a number  $i$  according to the number of nodes eliminated so far and denote the family by  $V_i$
- Denote the set of remaining nodes  $S_i$



# Triangulated, undirected Graph

## -> Join trees

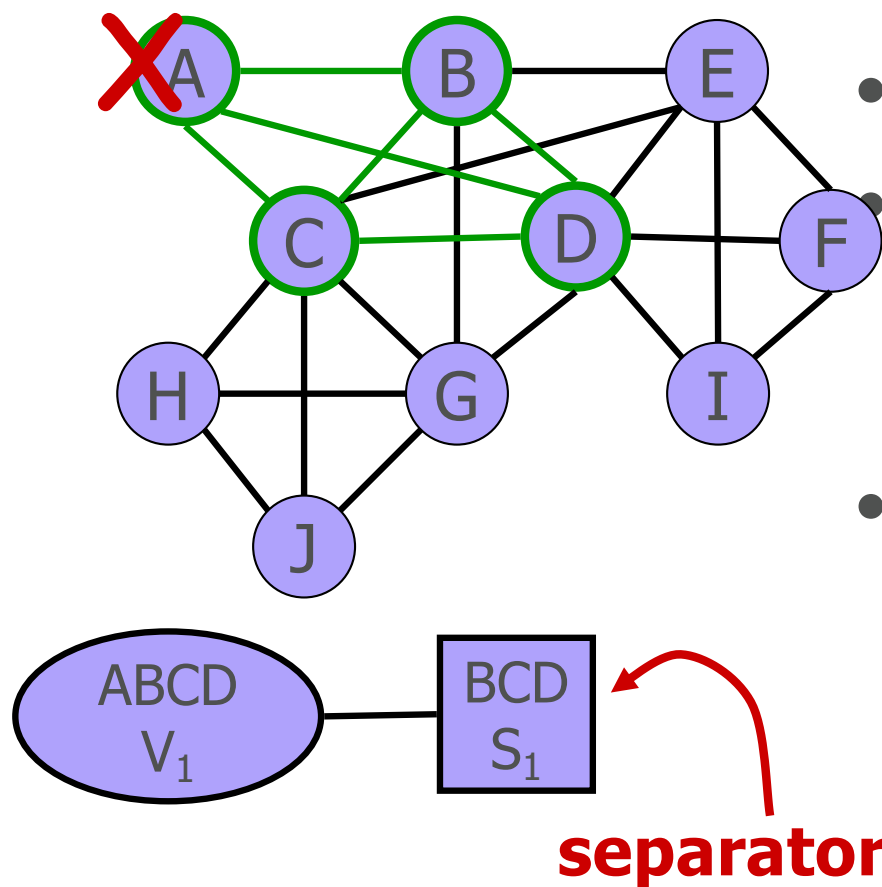


- Simplicial node  $X$
- Family of  $X$  is a clique
- Eliminate nodes from family of  $X$  which have only neighbours in the family of  $X$
- Give family of  $X$  a number  $i$  according to the number of nodes eliminated so far and denote the family by  $V_i$
- Denote the set of remaining nodes  $S_i$



# Triangulated, undirected Graph

## -> Join trees



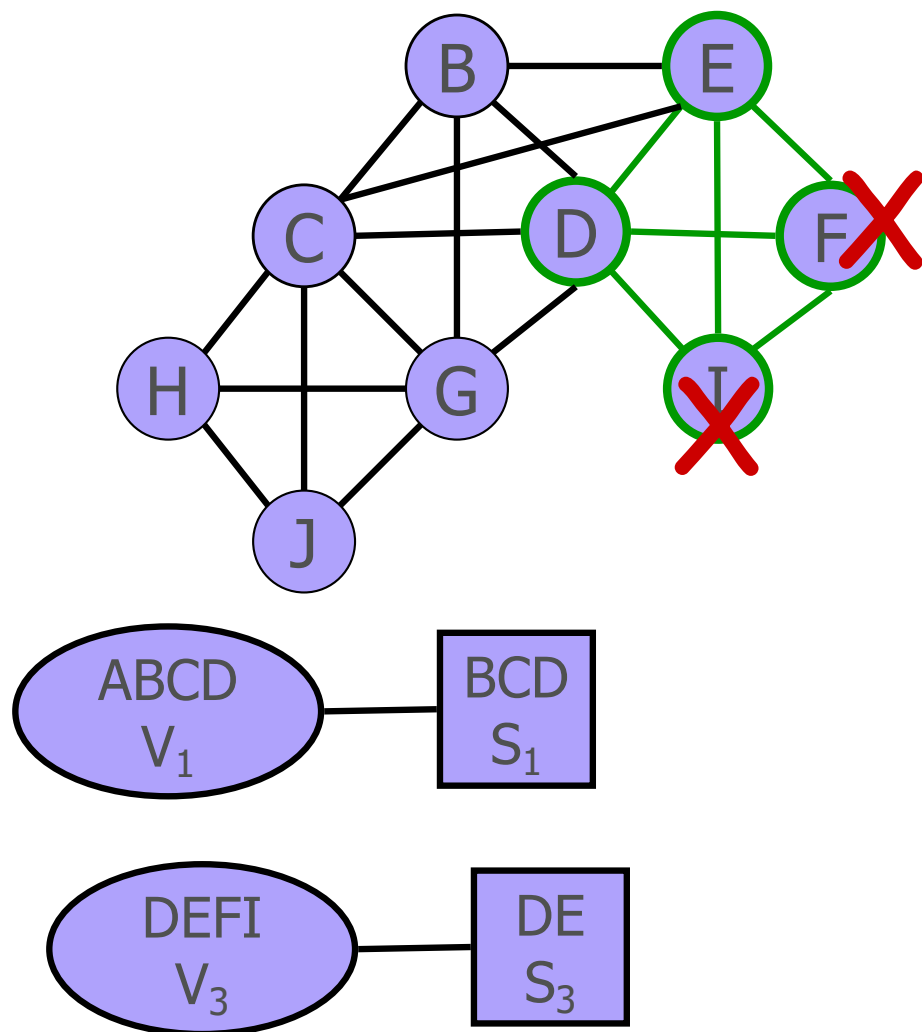
- Simplicial node X **A**
- Family of X is a clique **{A,B,C,D}**
- Eliminate nodes from family of X which have only neighbours in the family of X **{A}{B,C,D}**
- Give family of X a number i according to the number of nodes eliminated so far and denote the family by V<sub>i</sub> **V<sub>1</sub>**
- Denote the set of remaining nodes S<sub>i</sub> **S<sub>1</sub>**





# Triangulated, undirected Graph

## -> Join trees

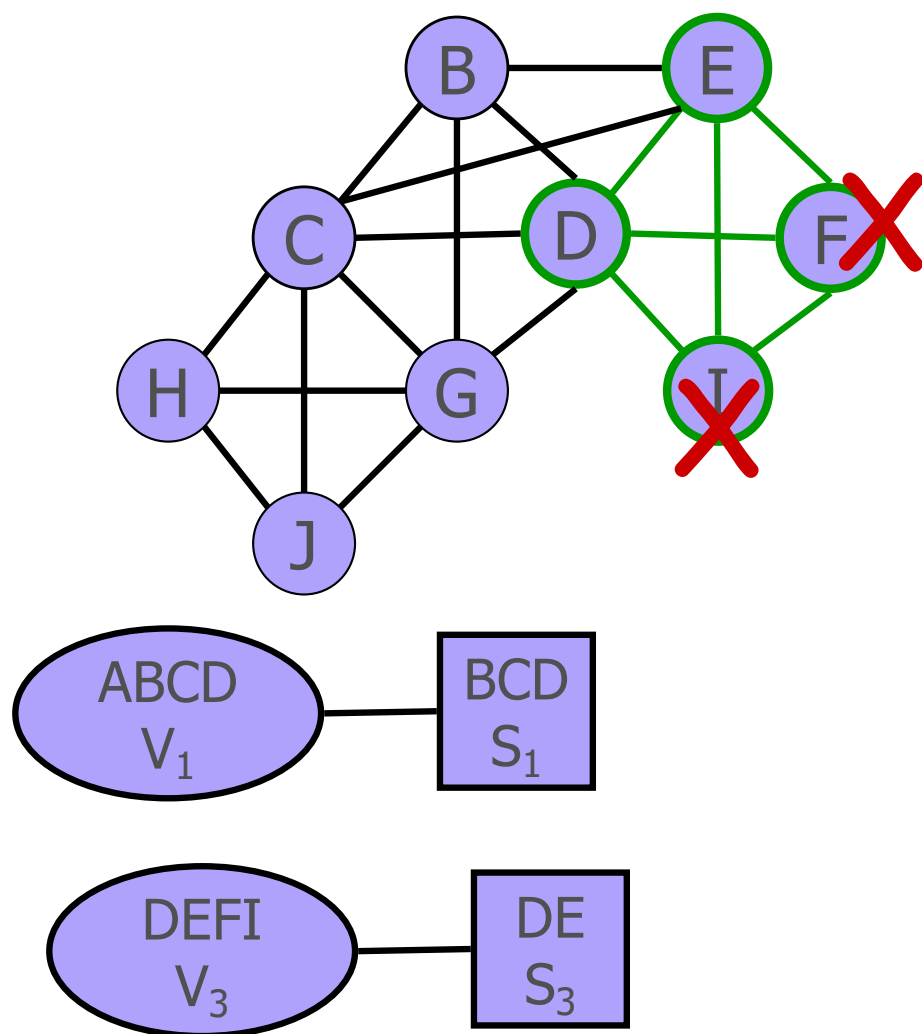


- Simplicial node X
- Family of X is a clique
- Eliminate nodes from family of X which have only neighbours in the family of X
- Give family of X a number i according to the number of nodes eliminated so far and denote the family by  $V_i$
- Denote the set of remaining nodes  $S_i$



# Triangulated, undirected Graph

## -> Join trees



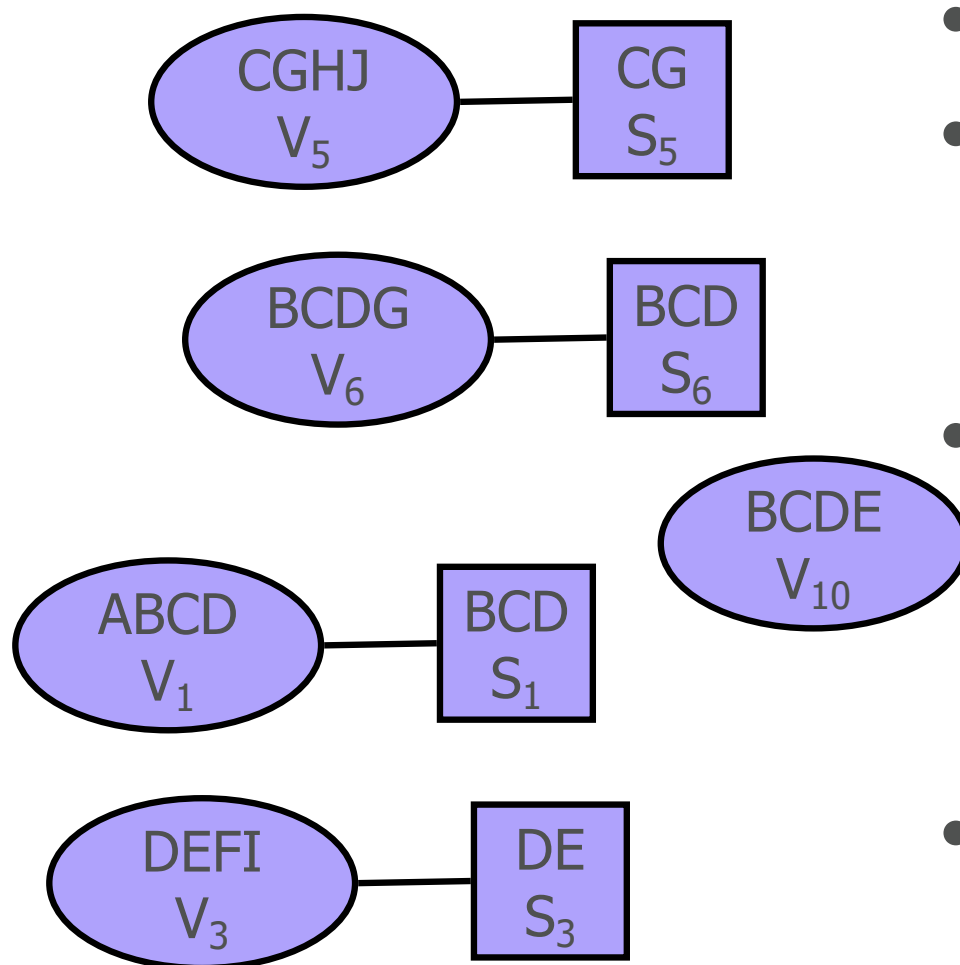
- Simplicial node X **F**
- Family of X is a clique **{D,E,F,I}**
- Eliminate nodes from family of X which have only neighbours in the family of X **{F,I}{D,E}**
- Give family of X a number i according to the number of nodes eliminated so far and denote the family by  $V_i$   **$V_3$**
- Denote the set of remaining nodes  $S_i$   **$S_3$**

• We stopped here



# Triangulated, undirected Graph

## -> Join trees

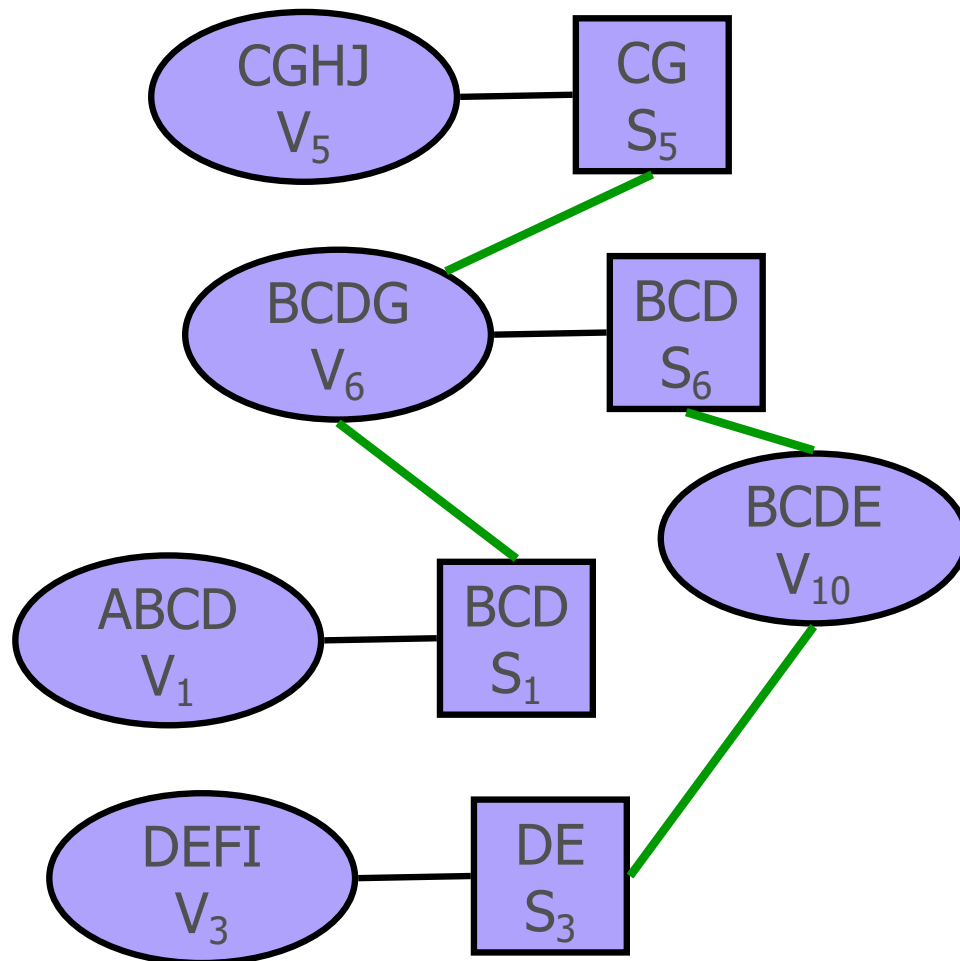


- Simplicial node  $X$
- Family of  $X$  is a clique
- Eliminate nodes from family of  $X$  which have only neighbours in the family of  $X$
- Give family of  $X$  a number  $i$  according to the number of nodes eliminated so far and denote the family by  $V_i$
- Denote the set of remaining nodes  $S_i$



# Triangulated, undirected Graph

## -> Join trees

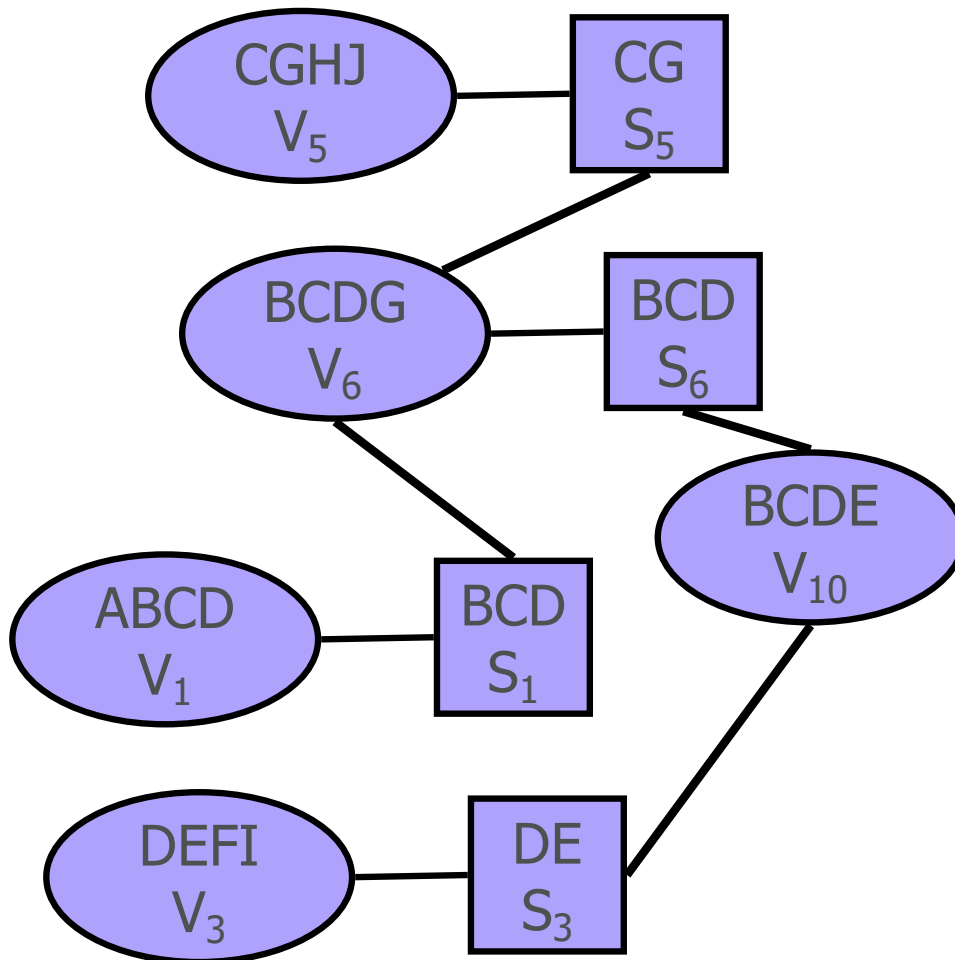


- Connect each separator  $S_i$  to a clique  $V_j$ ,  $j > i$ , such that  $S_i$  is a subset of  $V_j$
- Due to the running intersection property this is always possible



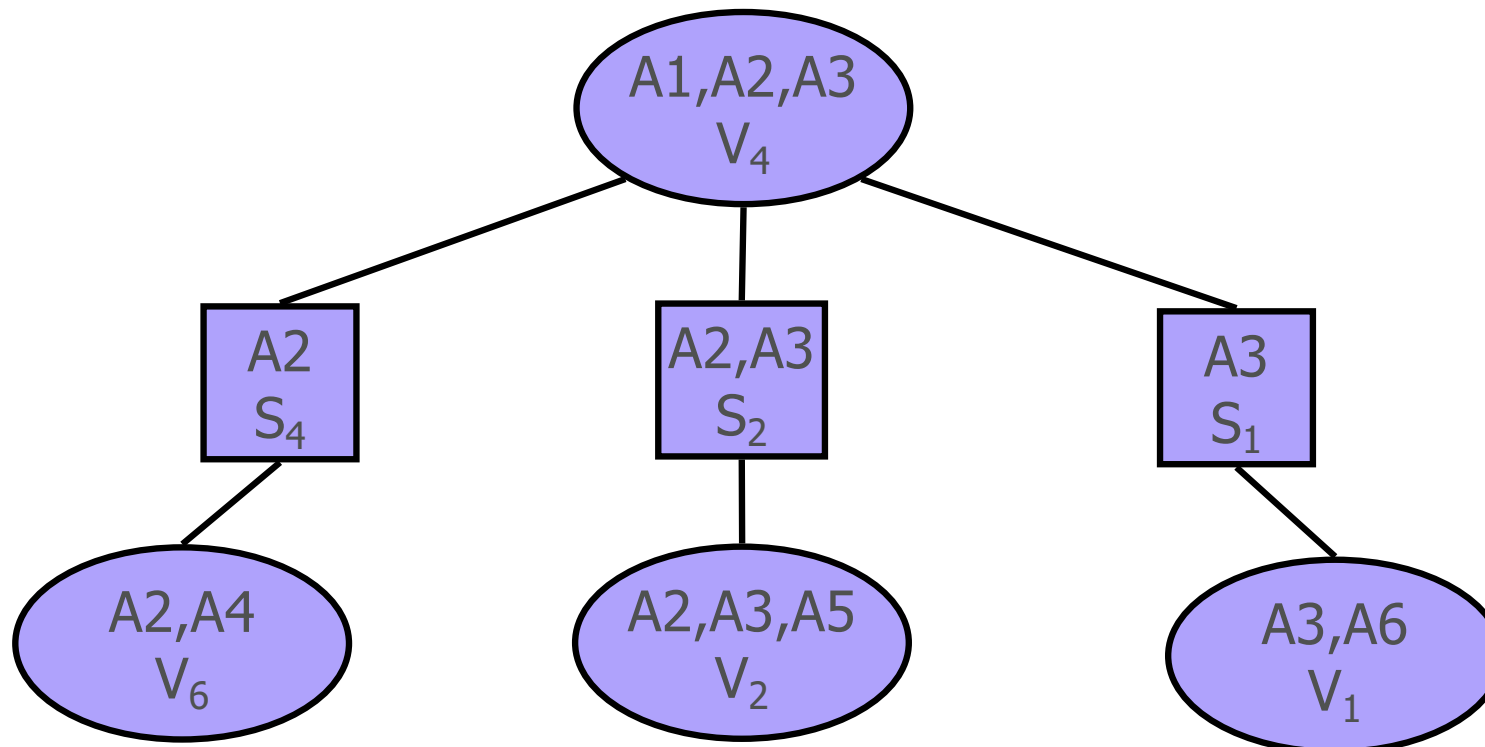
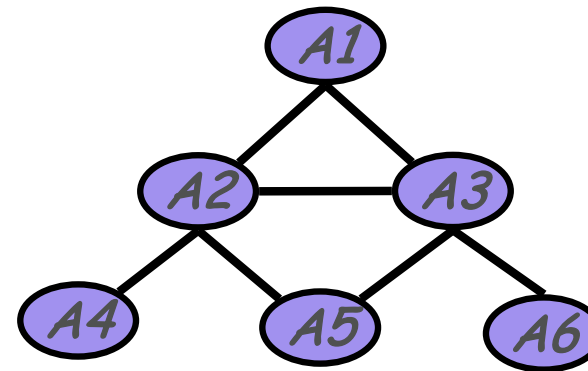
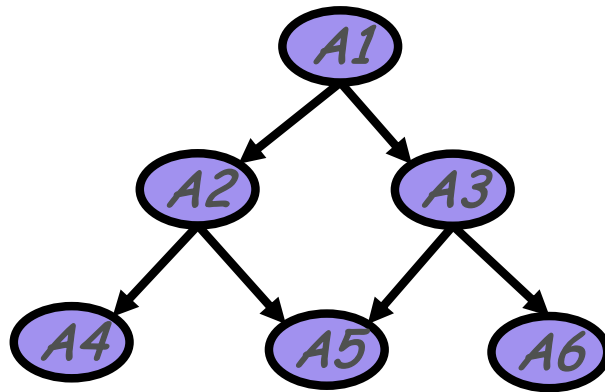
# Join Tree

The potential representation of a join tree (aka clique tree) is the product of the clique potentials, divided by the product of the separator potentials.



$$P(\mathbf{X}) = \frac{\prod_c \phi_c(\mathbf{X})}{\prod_s \phi_s(\mathbf{X})}$$

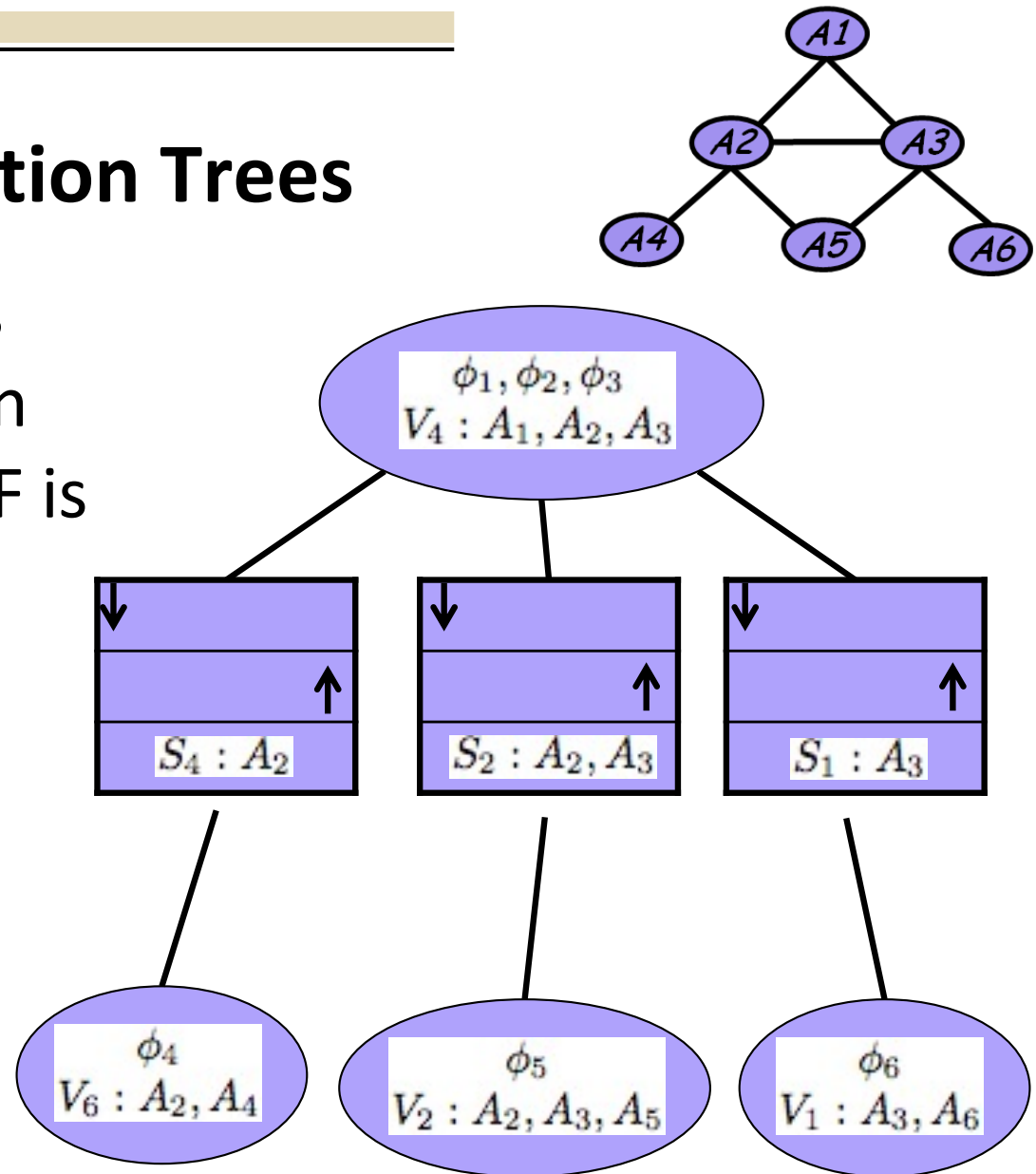
# Yet Another Example



# Junction Trees

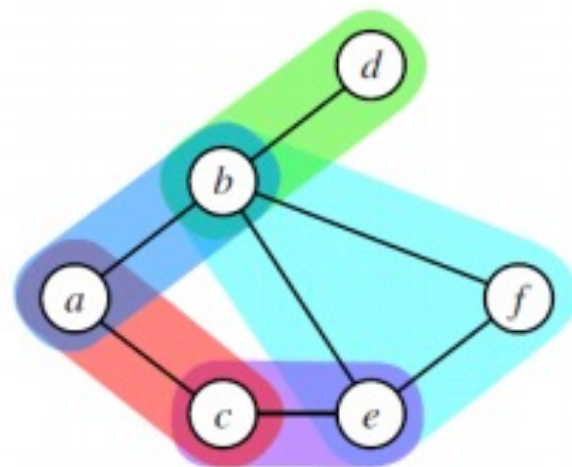
- Let  $F$  be a set of potentials with a triangulated domain graph. A junction tree for  $F$  is join tree for  $G$  with

- Each potential  $f$  in  $F$  is associated to a clique containing  $\text{dom}(f)$
- Each link has separator attached containing two mailboxes, one for each direction

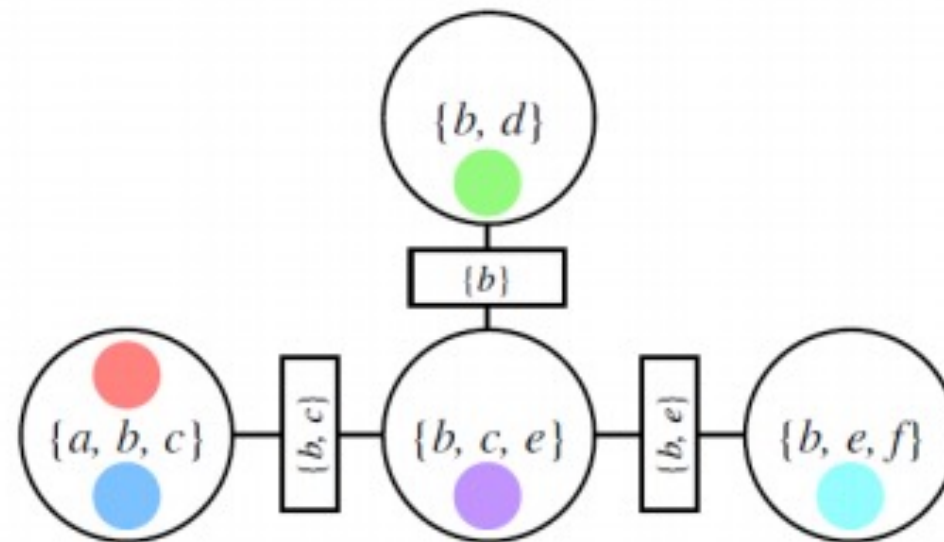


# Works naturally for Markov networks

*Family preservation:* For each factor  $f$ , there is a cluster  $c$  such that  $\text{dom}(f) \subseteq c$ .



$G$



$T$

credit: Mark Paskin



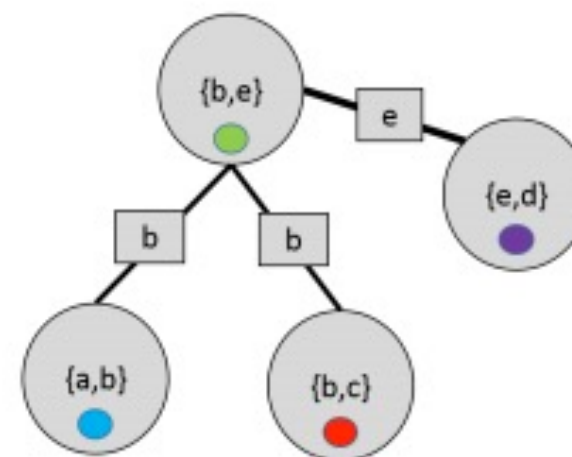
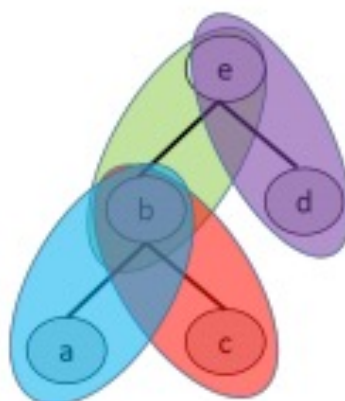
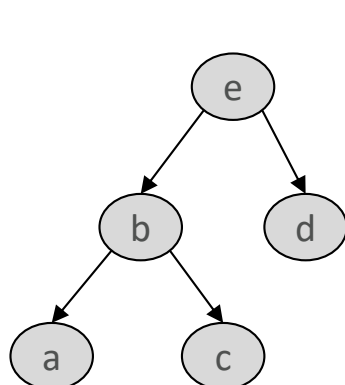
# Optimal Junction Trees

We may always find a **trivial junction tree** with one node containing all the variables in the original graph. However, such trees are useless because they will not result in efficient marginalization algorithms.

**Optimal trees** are one that make the clusters as small and modular as possible; unfortunately, it is again NP-hard to find the optimal tree. We will see below some practical ways in which we can find good junction trees.

# Junction tree of tree-structure BN

A special case when we can find the optimal junction tree is when the model itself is a tree. Then, we may define a cluster for each edge in the tree.

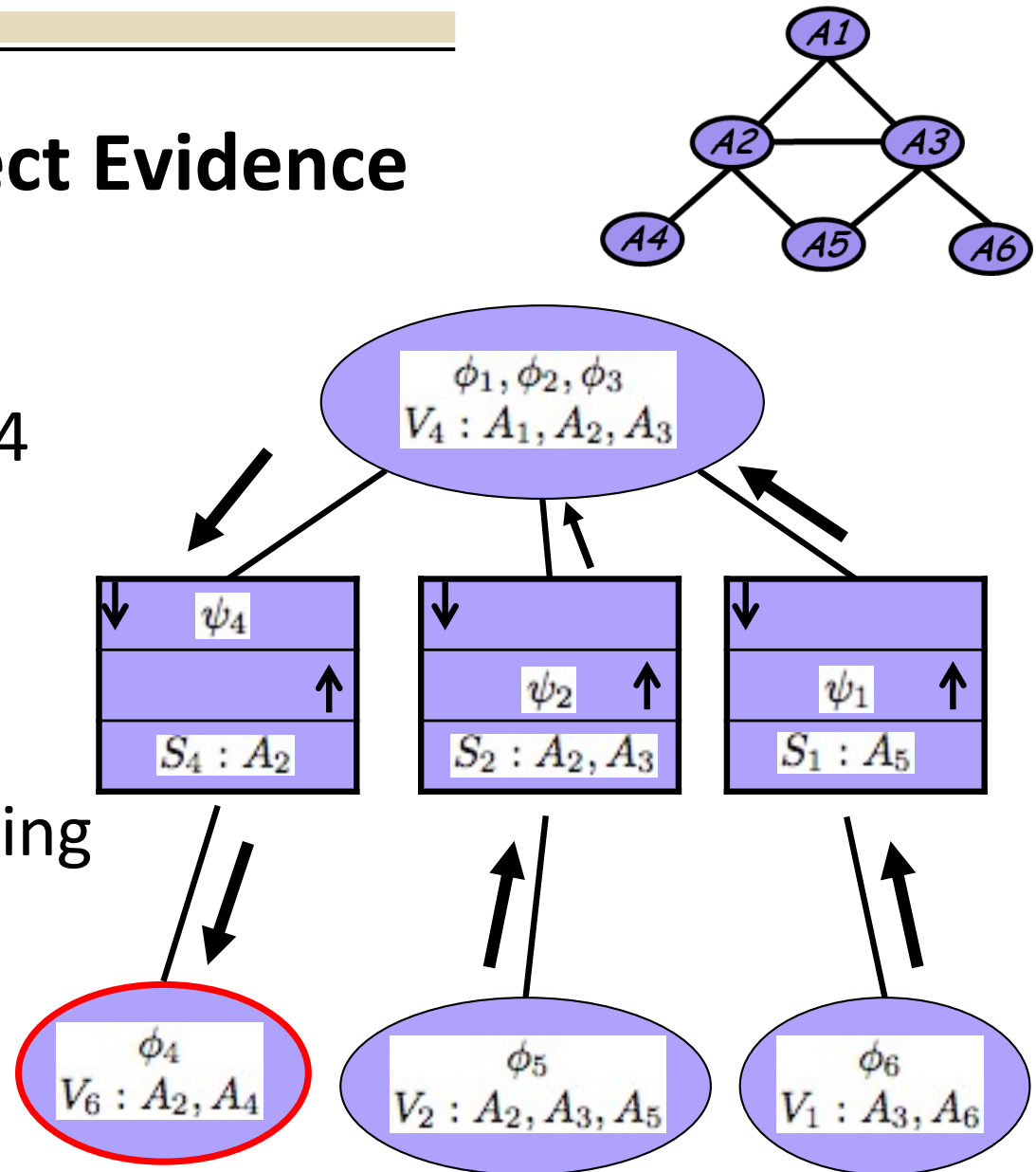


# Propagation on a Junction Tree

- Node  $V$  can send exactly one message to a neighbour  $W$ , and it may only be sent when  $V$  has received a message from all of its other neighbours
- Choose one clique (arbitrarily) as a root of the tree; collect message to this node and then distribute messages away from it.
- After collection and distribution phases, we have all we need in each clique to compute potential for variables.

# Junction Trees - Collect Evidence

- $P(A_4)$  ?
- Find clique containing  $A_4$
- $V_6$  temporary root
- Send messages from leaves to root
- $V_4$  assembles the incoming messages, potential



## Junction Tree (for Bayesian networks) - Messages

- Propagation/message passing between two adjacent cliques  $C_1, C_2$  ( $S_0$  is their separator)

- Marginalize  $C_1$ 's potential to get new potential for  $S_0$

$$\varphi_{S_0}^* = \sum_{C_1 \setminus S_0} \varphi_{C_1}$$

- Update  $C_2$ 's potential

$$\varphi_{C_2}^* = \varphi_{C_2} \frac{\varphi_{S_0}^*}{\varphi_{S_0}}$$

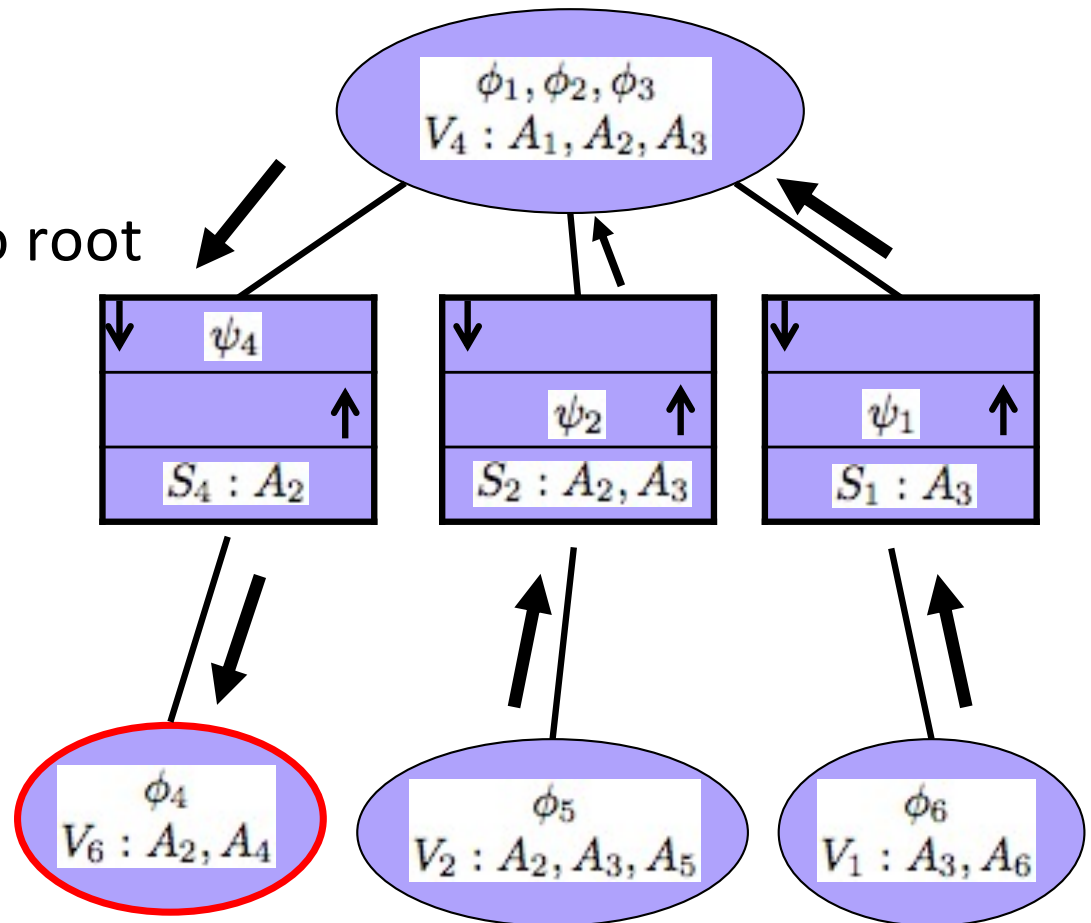
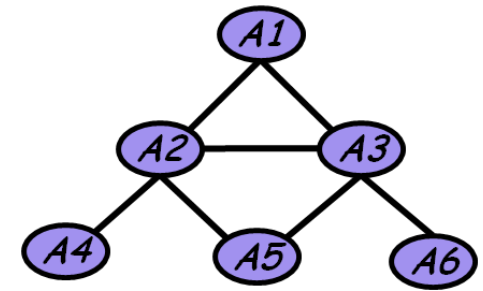
- Update  $S_0$ 's potential to its new potential. **Initially, its potential is 1**, i.e.,  $\varphi_{C_2}^* = \varphi_{C_2} \varphi_{S_0}^*$

- **That is, we sent a message  $\varphi_{S_0}^*$  from  $C_1$  to  $C_2$**



# Junction Trees - Collect Evidence

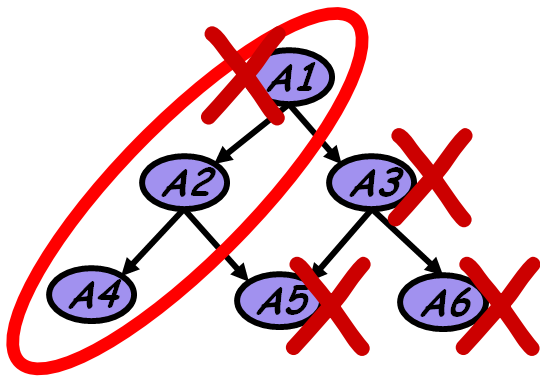
- $P(A_4)$  ?
- Find clique containing  $A_4$
- $V_6$  temporary root
- Send messages from leaves to root
- $V_4$  assembles the incoming messages, potential



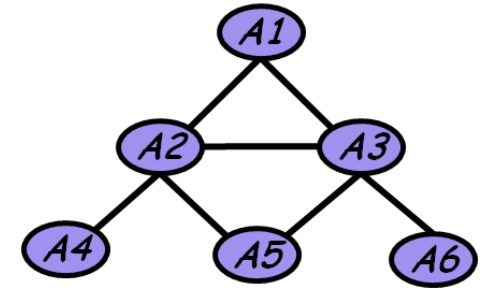
$$\varphi_1 = \sum_{V_1 - \{A_3\}} \phi_6 = \sum_{A_6} \phi_6$$

$$\varphi_2 = \sum_{V_2 - \{A_2, A_3\}} \phi_5 = \sum_{A_5} \phi_5$$

$$\psi_4 = \sum_{A_1} \phi_1 \cdot \phi_2 \sum_{A_3} \phi_3 \cdot \psi_2 \cdot \psi_1$$



**This is VE !**



First, we have

$$\varphi_1 = \sum_{V_1 - \{A_3\}} \phi_6 = \sum_{A_6} P(A_6 | A_3) = 1$$

$$\varphi_2 = \sum_{V_1 - \{A_2, A_3\}} \phi_5 = \sum_{A_5} P(A_5 | A_2, A_3) = 1$$

**So, we have eliminated A5 and A6**

Then

$$\varphi_4 = \sum_{A_1} \phi_1 \cdot \phi_2 \cdot \sum_{A_3} \phi_3 \cdot \varphi_1 \cdot \varphi_2 = \sum_{A_1} \phi_1 \cdot \phi_2 \cdot \sum_{A_3} \phi_3 = \sum_{A_1} \phi_1 \cdot \phi_2 \cdot 1$$

**So, we have eliminated A3 and are in the „chain“ situation. First, we eliminate A1 and then ...**



# Junction Trees - Collect Evidence

- $P(A_4)$  ?

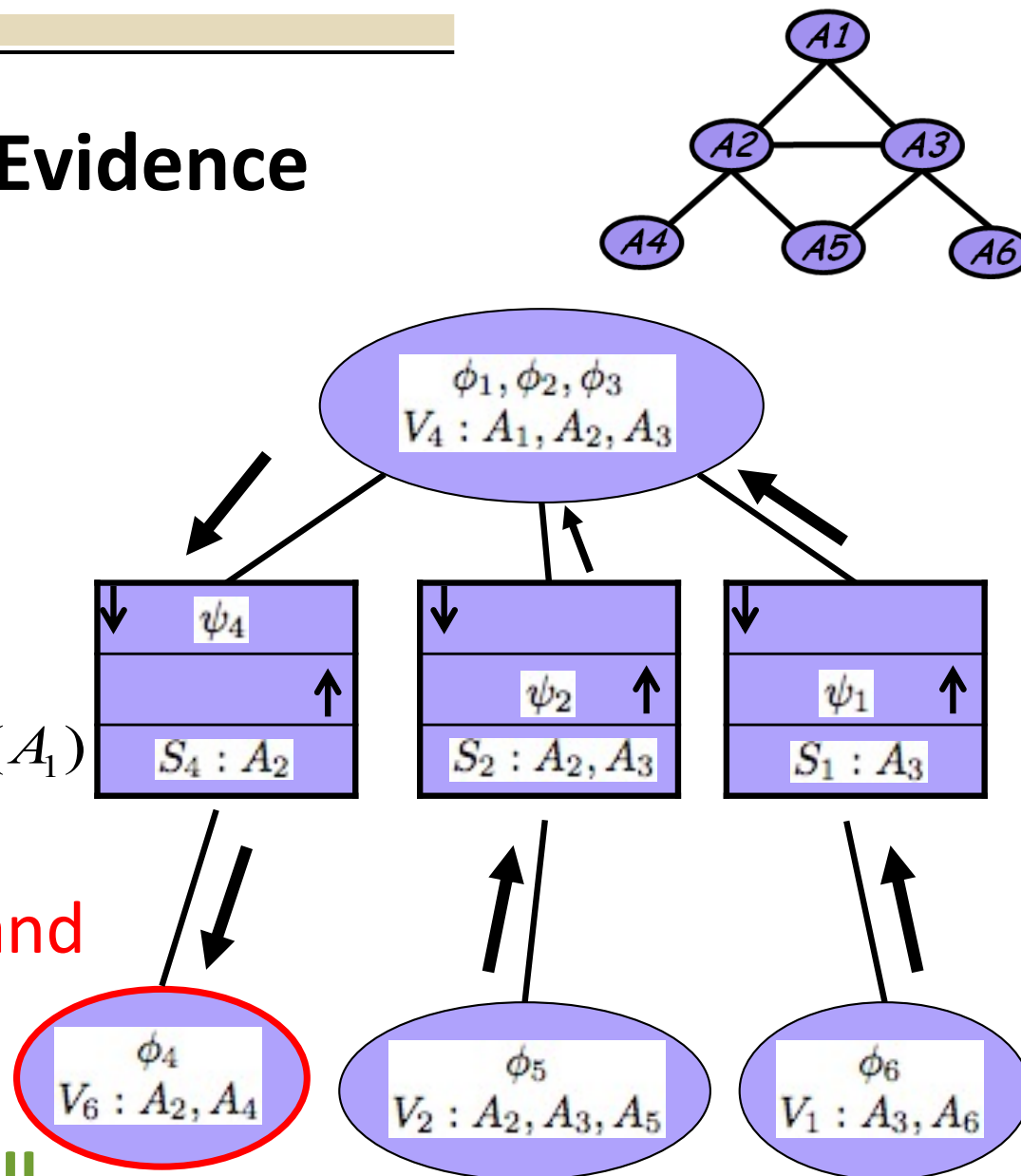
$$P(A_4) = \sum_{A_2} \psi_4 \cdot \phi_4$$

$$P(A_4) = \sum_{A_2} \phi_4 \cdot \sum_{A_1} \phi_1 \cdot \phi_2$$

$$= \sum_{A_2} P(A_4 | A_2) \cdot \sum_{A_1} P(A_2 | A_1) \cdot P(A_1)$$

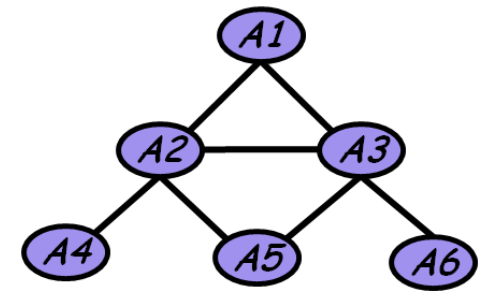
- That is, we eliminate  $A_2$  and get  $P(A_4)$

- What about computing all marginals?



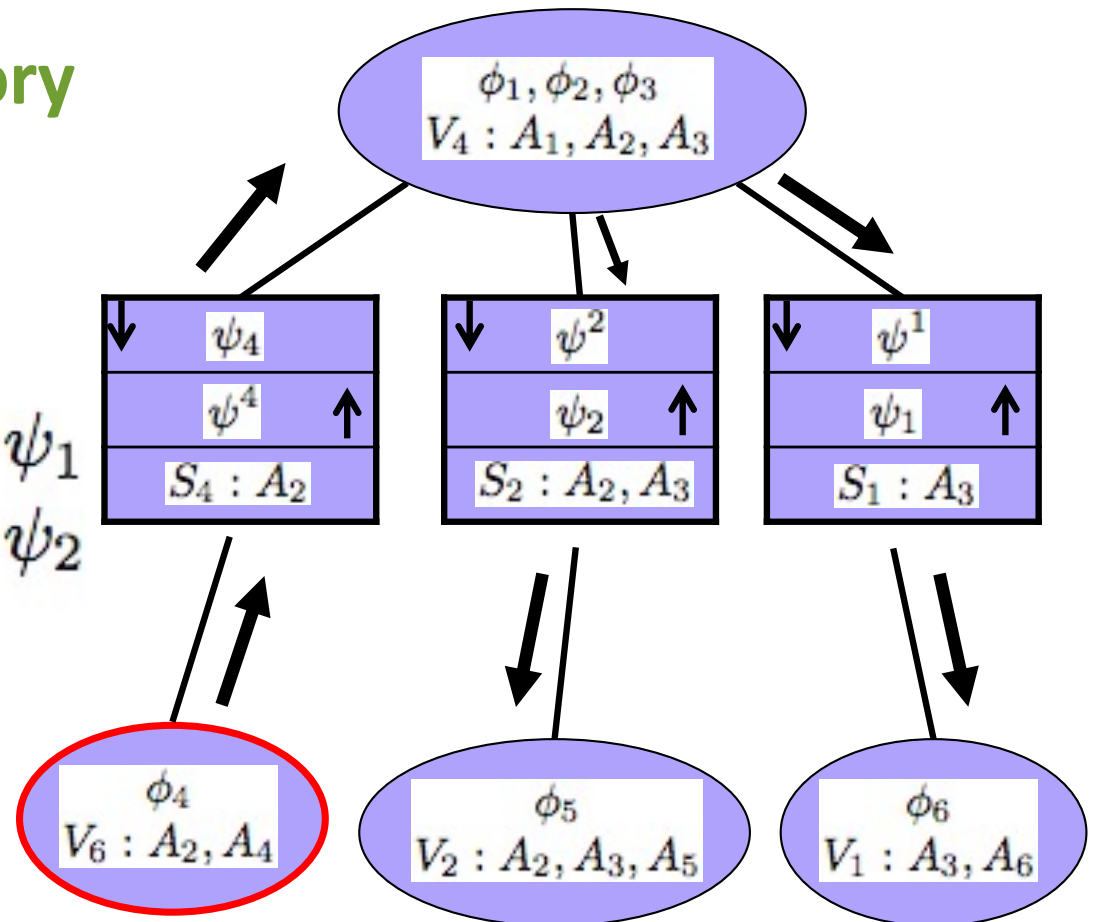


# Junction Trees - Distribute Evidence



- All marginals? **Same story**

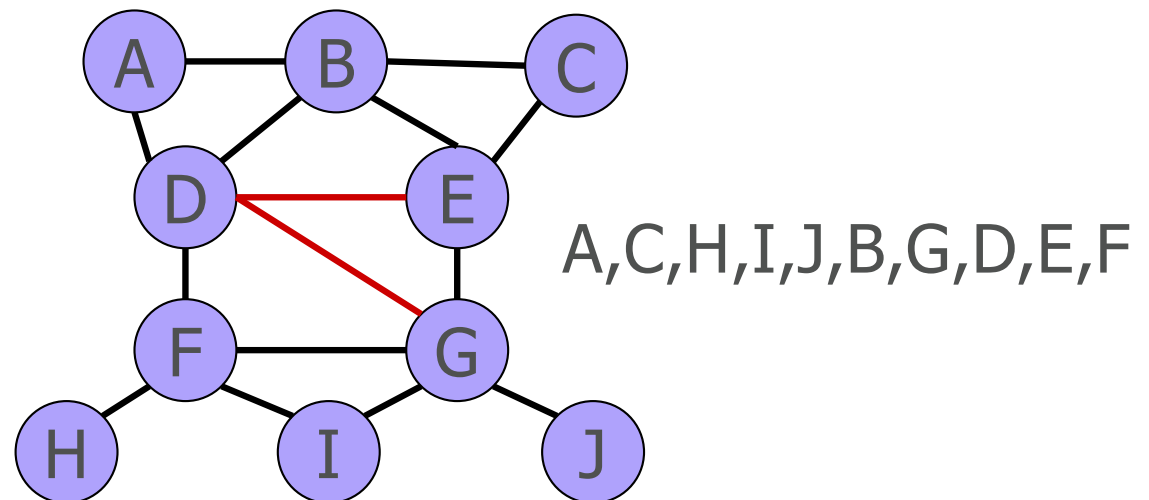
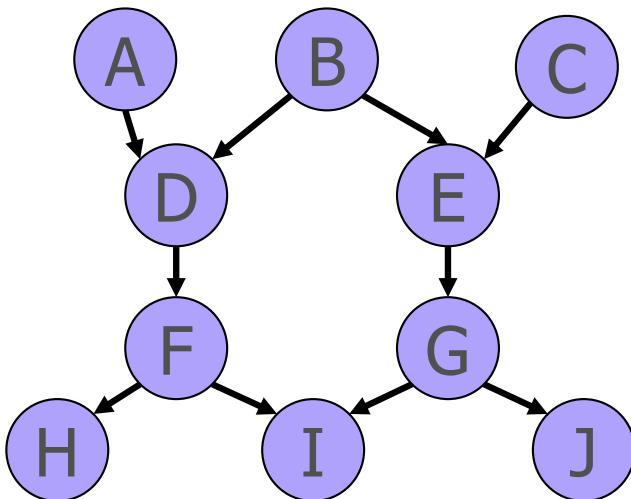
$$\begin{aligned}\psi^4 &= \sum_{A_2} \phi_4 \\ \psi^2 &= \psi^4 \cdot \left( \sum_{A_1} \phi_1 \phi_2 \phi_3 \right) \cdot \psi_1 \\ \psi^1 &= \psi^4 \cdot \left( \sum_{A_1} \phi_1 \phi_2 \phi_3 \right) \cdot \psi_2 \\ P(A_3) &= \sum_{A_6} \phi_6 \cdot \psi^1 \\ P(A_6) &= \sum_{A_3} \phi_6 \cdot \psi^1 \\ \dots\end{aligned}$$



Same can be done for undirected models. Separators also account for their own potentials

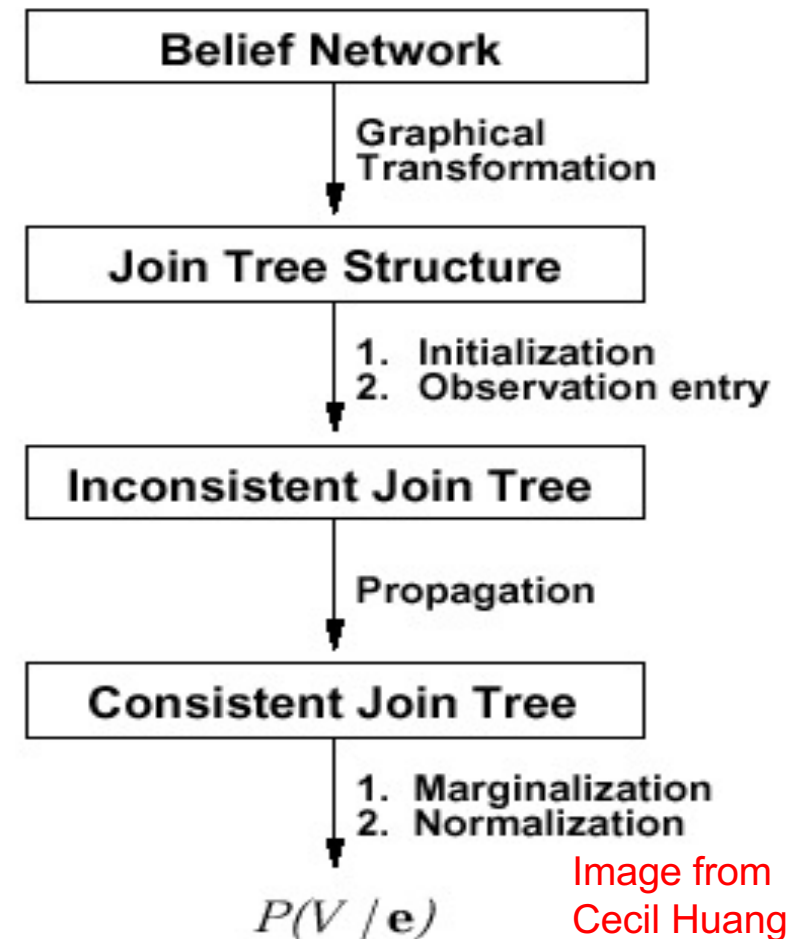
# Nontriangulated Domain Graphs

- Embed domain graph in a triangulated graph
- Use its junction tree
- Simple idea:
  - Eliminate variables in some order
  - If you wish to eliminate a node with non-complete neighbour set, make it complete by adding fill-ins



# Summary JTA

- Convert Bayesian network into JT
- Initialize potentials and separators
- Incorporate Evidence (set potentials accordingly)
- Collect and distribute evidence
- Obtain clique marginals by marginalization/normalization



# Inference Engines

- (Commercial) HUGIN : <http://www.hugin.com>
- (Commercial) NETICA: <http://www.norsys.com>
- Bayesian Network Toolbox for Matlab  
[www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html](http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html)
- GENIE/SMILE (JAVA) <http://www2.sis.pitt.edu/~genie/>
- MSBNx, Microsoft, <http://research.microsoft.com/adapt/MSBNx/>
- LibDAI <http://people.kyb.tuebingen.mpg.de/jorism/libDAI/>
- OpenGM2 <http://hciweb2.iwr.uni-heidelberg.de/opengm/>