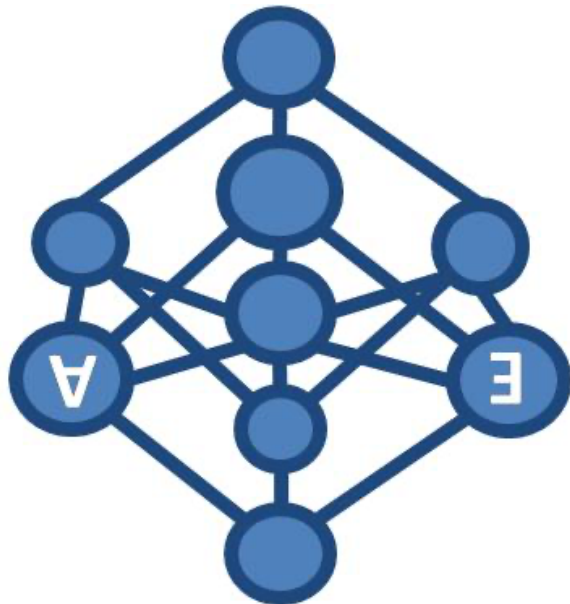


# Probabilistic Graphical Models\*

## Bayesian Networks



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



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



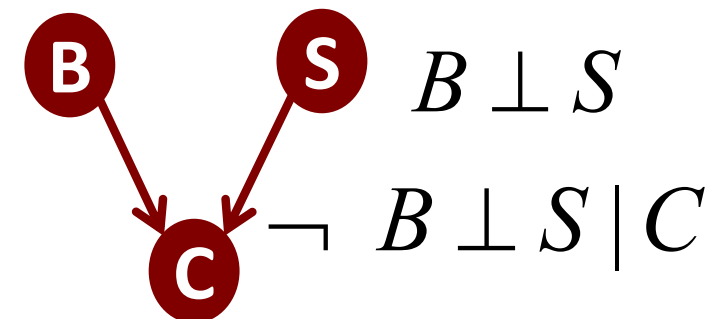
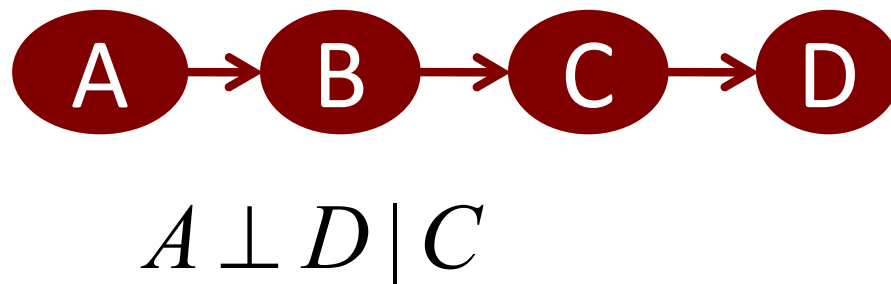
# What you need to know thus far

- Independence and conditional independence
- Definition of a Bayesian network
- Local Markov assumption
- The representation theorem
  - $G$  is I-map for  $P$  iff  $P$  factorizes according to  $G$
  - Interpretation



# Independencies encoded in BN

- We said: all you need is the local Markov assumption ( $X_i \perp \text{NonDescendants}_{X_i} \mid \text{Pa}_{X_i}$ )
- But then we talked about other (in)dependencies such as explaining away



- **So, what are the independencies encoded by a BN?**
  - Only assumption is local Markov but many other independencies can be derived using the algebra of conditional independencies!

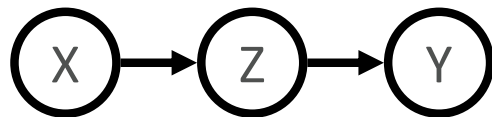


# Understanding independencies in BNs (with 3 nodes)

Local Markov Assumption: A variable  $X$  is independent of its non-descendants given its parents and only its parents:

$$(X_i \perp \text{NonDescendants}_{X_i} \mid \text{Pa}_{X_i})$$

Indirect causal effect:



$$X \perp Y \mid Z$$

$$\neg X \perp Y$$

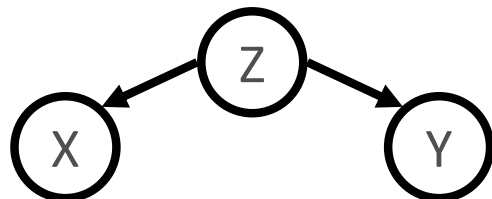
Indirect evidential effect:



$$X \perp Y \mid Z$$

$$\neg X \perp Y$$

Common cause:

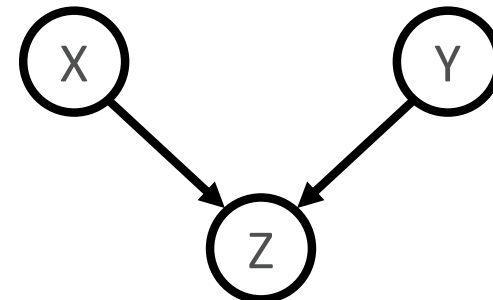


$$X \perp Y \mid Z$$

$$\neg X \perp Y$$

Represent all the same distributions

(v-structure)  
Common effect:



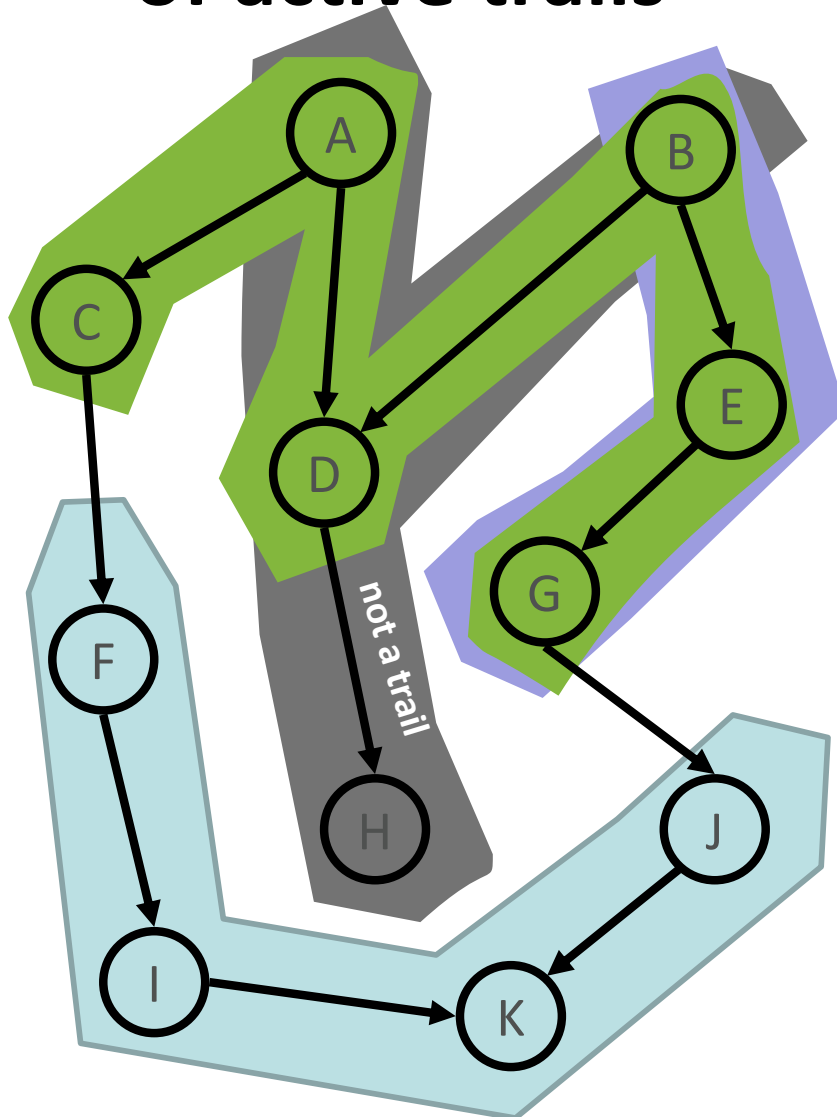
*inverted*

$$X \perp Y$$

$$\neg X \perp Y \mid Z$$



# This can be generalized using the notion of active trails



A trail is an undirected path that never visits a node twice



# This can be generalized using the notion of active trails

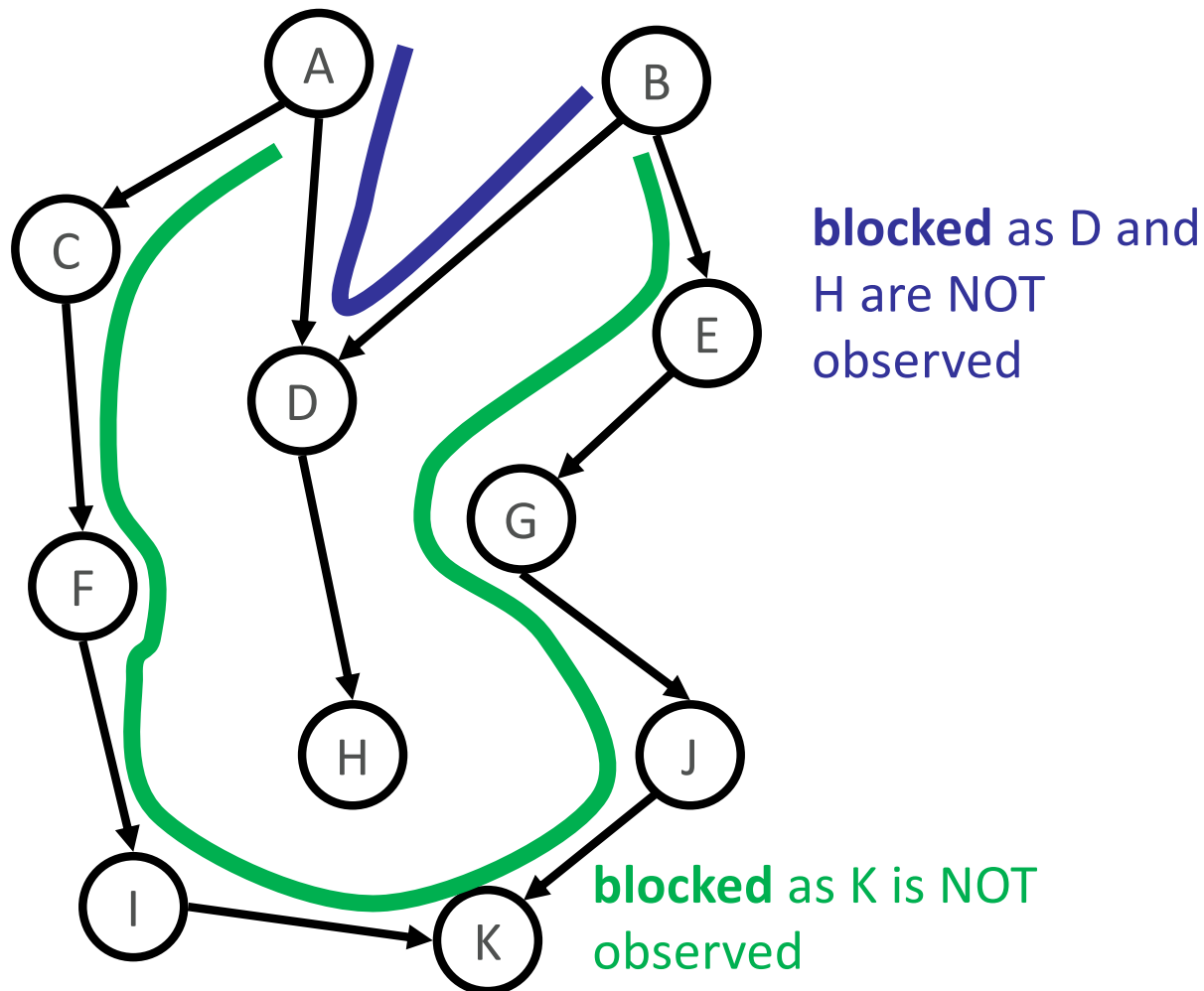
- A trail  $X'_1 - X'_2 - \dots - X'_k$  is **active** (when some variables  $\mathbf{O} \subseteq \{X_1, \dots, X_m\}$  are observed) **if** for each consecutive triplet in the trail it holds:
  - $X_{i-1} \rightarrow X_i \rightarrow X_{i+1}$ , and  $X_i$  is **not observed** ( $X_i \notin \mathbf{O}$ )
  - $X_{i-1} \leftarrow X_i \leftarrow X_{i+1}$ , and  $X_i$  is **not observed** ( $X_i \notin \mathbf{O}$ )
  - $X_{i-1} \leftarrow X_i \rightarrow X_{i+1}$ , and  $X_i$  is **not observed** ( $X_i \notin \mathbf{O}$ )
  - $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ , and  $X_i$  **is observed** ( $X_i \in \mathbf{O}$ ), or one of its **descendants (v-structure)**

**Intuitively, information flows along active trails!**



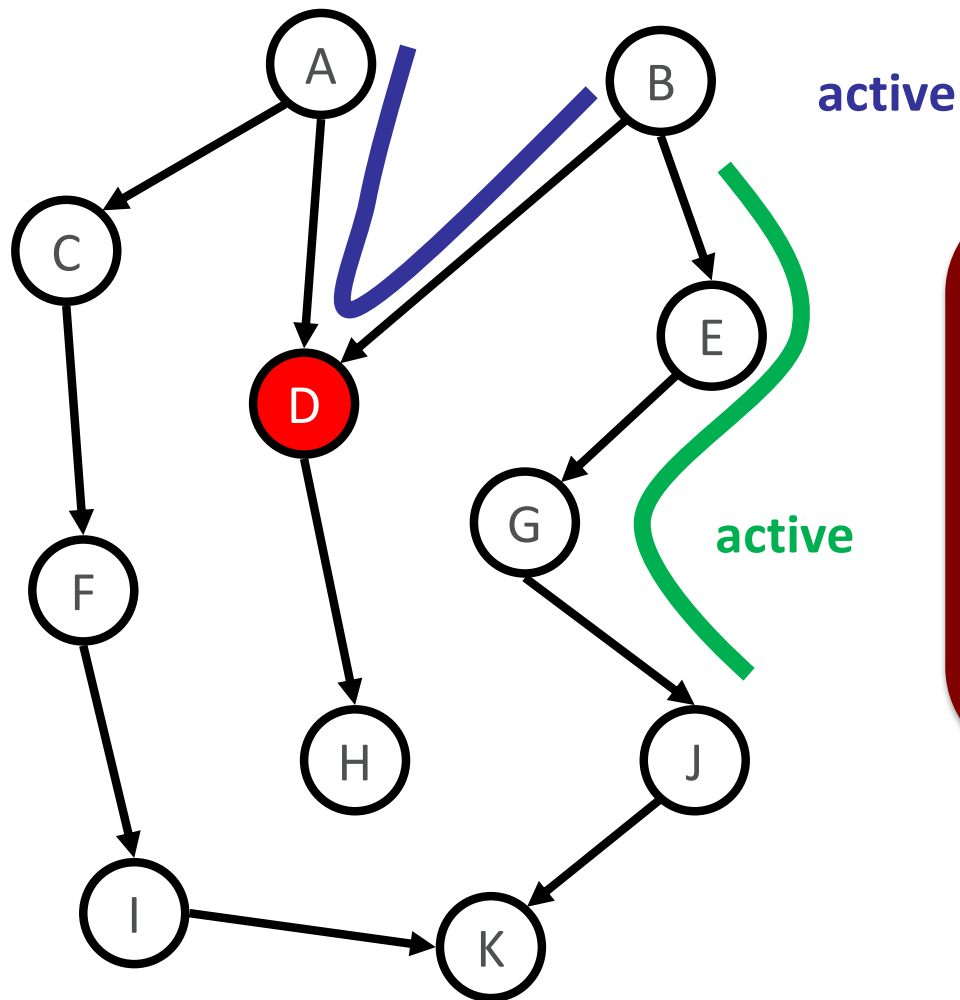
# Active and Blocked Trails in BNs

## Some Examples –



# Active and Blocked Trails in BNs

## Some Examples –

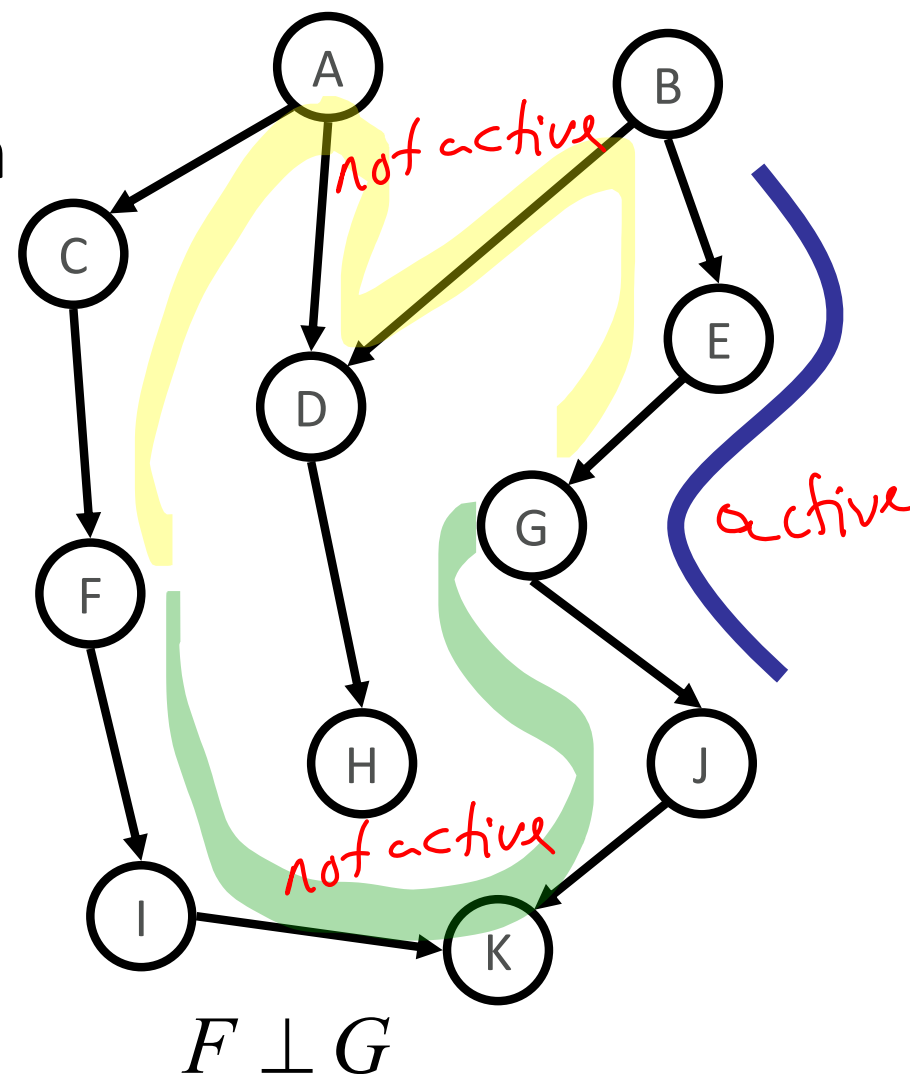


Information can flow if  
there is a trail  $x...y$  that  
is not blocked by  $z$   
(active trail)

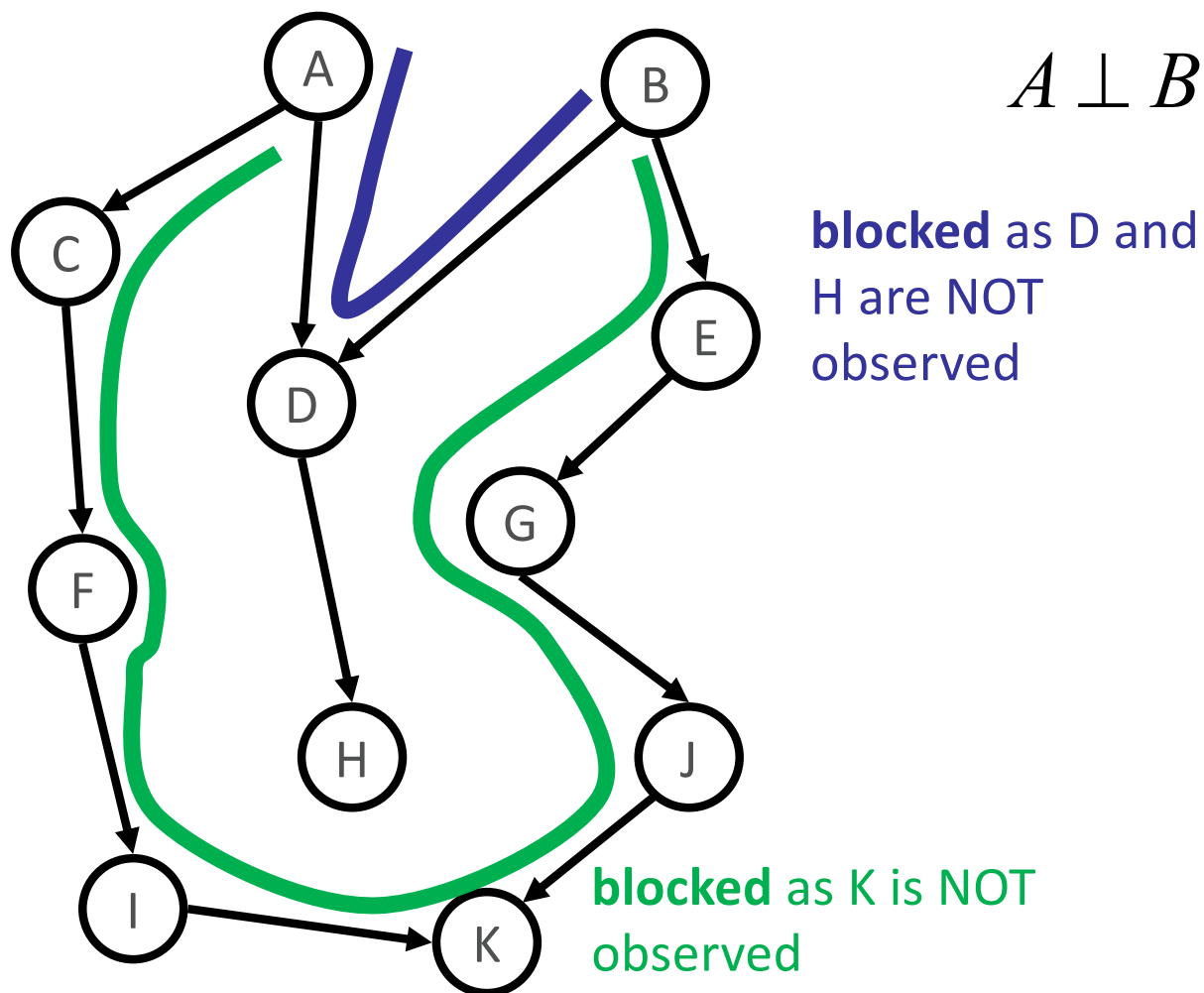


# Active trail and independence?

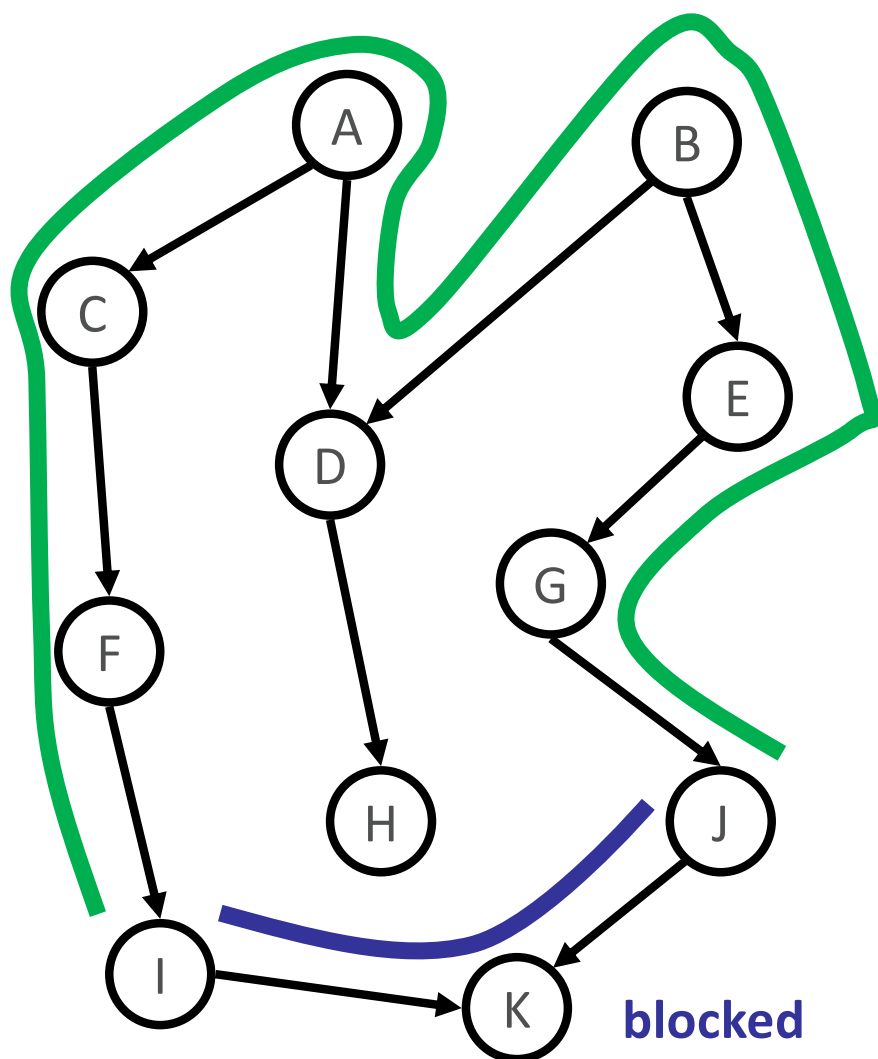
- **Theorem:** Sets of variables  $X_i$  and  $X_j$  are independent given  $Z \setminus \{X_1, \dots, X_n\}$  if there is **no active trail** between  $X_i$  and  $X_j$  when variables  $Z \setminus \{X_1, \dots, X_n\}$  are observed
- We say that  $X_i$  and  $X_j$  are **d-separated** given  $Z$  (dependence separation)



# D-Separation – Some Examples



# D-Separation – Some Examples



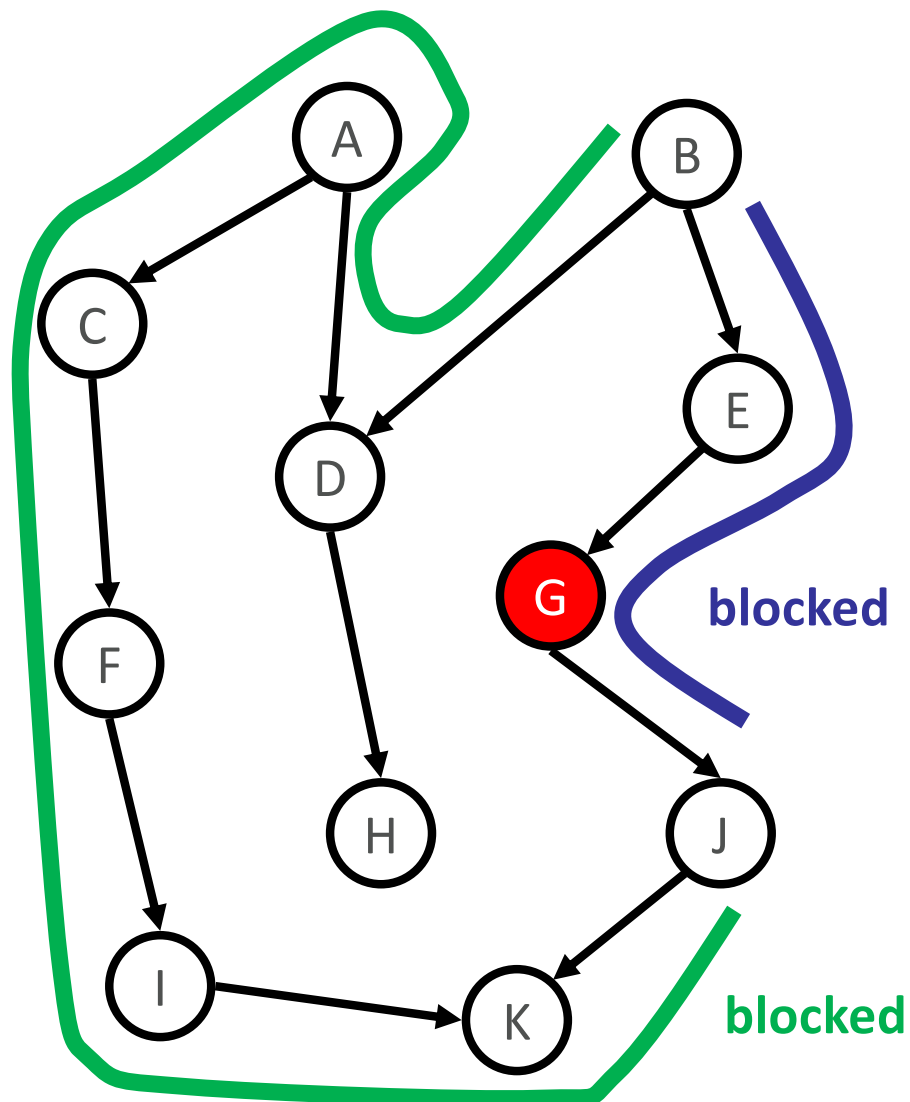
$$A \perp B \quad I \perp J$$

blocked

blocked



# D-Separation – Some Examples

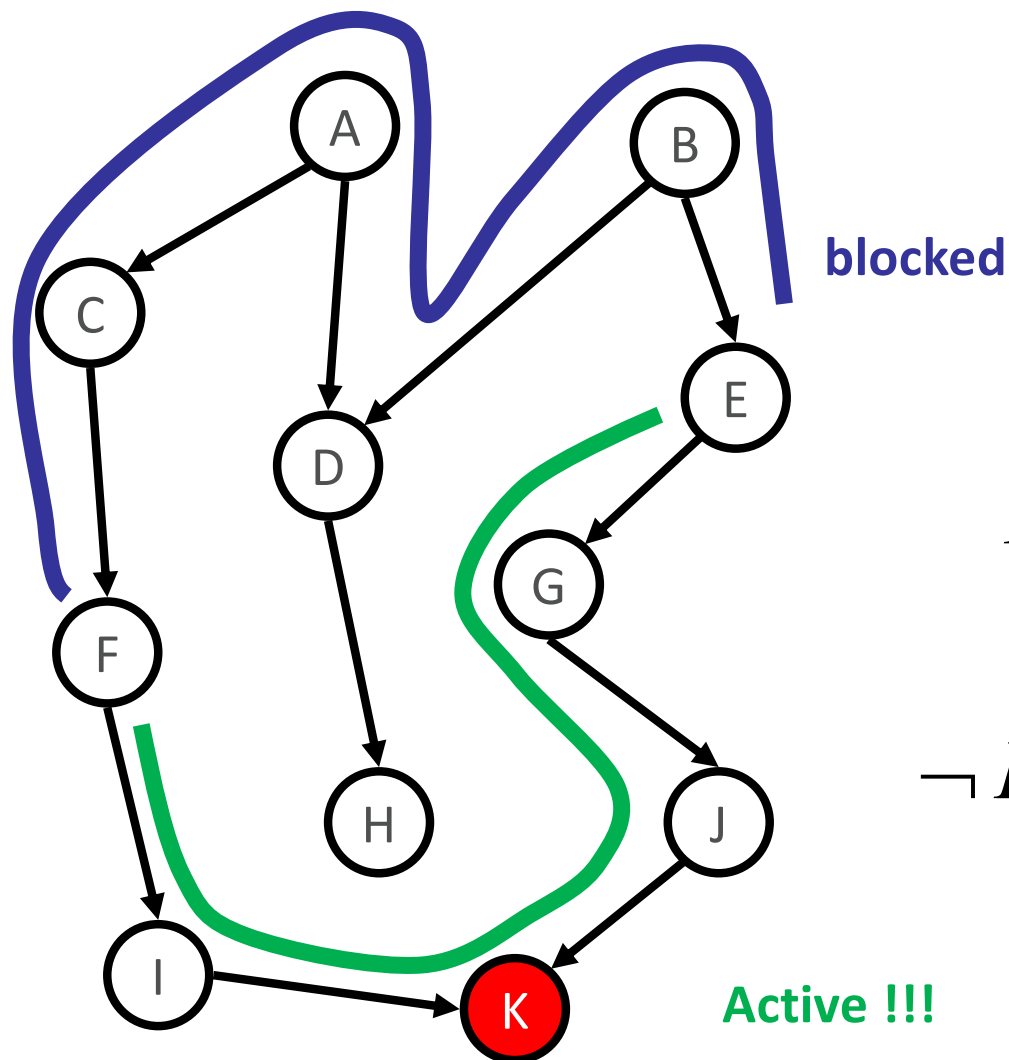


$$A \perp B \quad I \perp J$$

$$B \perp J \mid G$$



# D-Separation – Some Examples



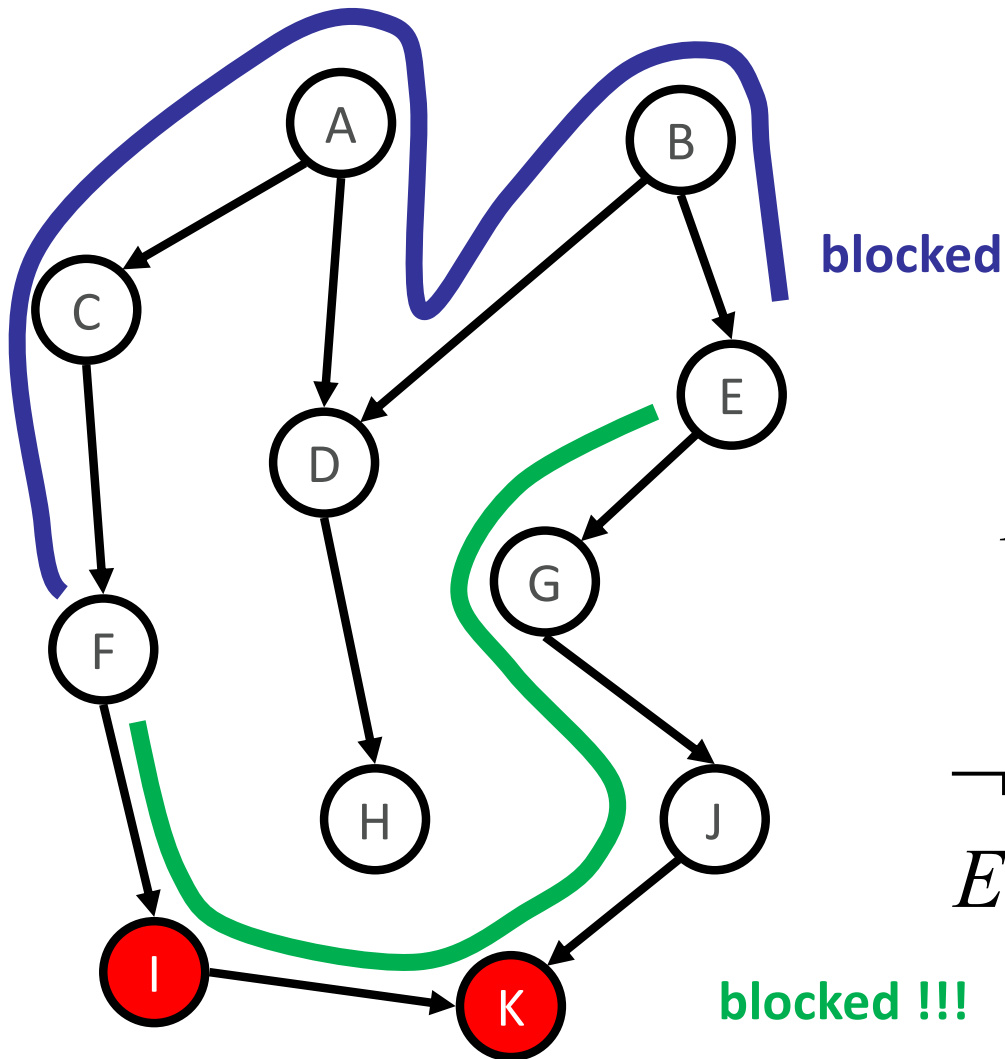
$$A \perp B \quad I \perp J$$

$$B \perp J \mid G$$

$$\neg E \perp F \mid K$$



# D-Separation – Some Examples



$$A \perp B \quad I \perp J$$

$$B \perp J \mid G$$

$$\neg E \perp F \mid K$$

$$E \perp F \mid K, I$$

The difference is  
 $E \perp F \mid I$



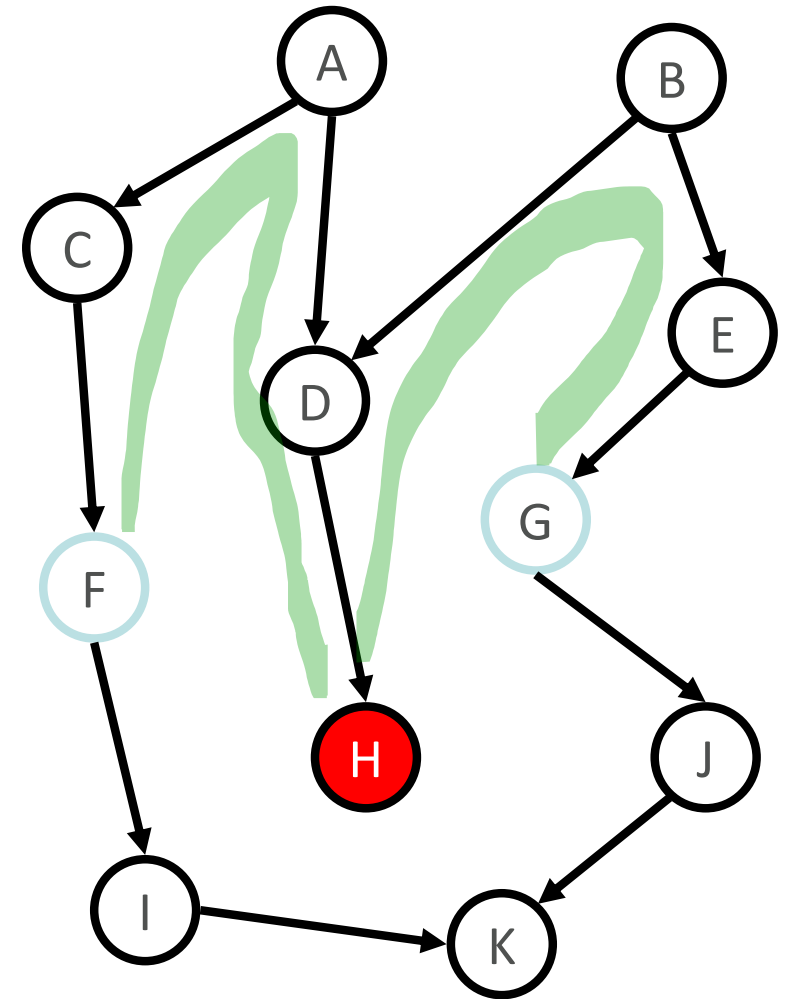
## More generally: Soundness of d-separation

- Given BN structure  $G$
- Set of independence assertions obtained by d-separation:  $I(G) = \{(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) : \text{d-sep}_G(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})\}$
- **Theorem: Soundness of d-separation**
  - If  $P$  factorizes over  $G$  then  $I(G) \subseteq I(P)$  and not only  $I_1(G) \subseteq I(P)$
- That means **d-separation only captures true independencies**



# Existence of dependency when not d-separated

- **Theorem:** If  $X$  and  $Y$  are **not** d-separated given  $\mathbf{Z}$ , then  $X$  and  $Y$  are dependent given  $\mathbf{Z}$  under **some  $P$**  that factorizes over  $G$
- **Proof sketch:**
  - Choose an active trail between  $X$  and  $Y$  given  $\mathbf{Z}$
  - Make this trail dependent
  - Make all else uniform (independent) to avoid “canceling” out influence



d-sep says

$$\neg F \perp G \mid H$$



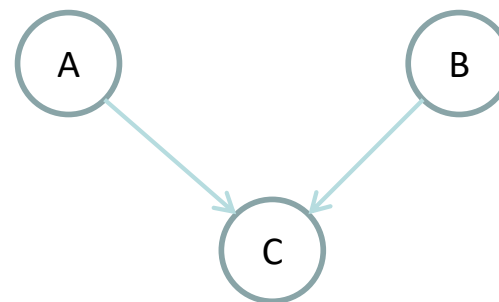


## Excursion: Faithful Distribution

- P is said to be **faithful** if it **does not** declare extra **independence** assumption that cannot be read from G

False	True
0.8	0.2

False	True
0.8	0.2



AB\C	False	True
FF	0.9	0.1
FT	0.9	0.1
TF	0.9	0.1
TT	0.9	0.1

P entails  $C \perp A, B$  !!

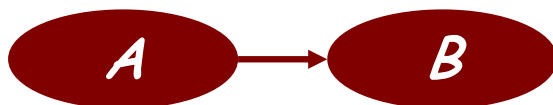
P is **not** faithful



## More generally: Completeness of d-separation

- **Theorem: Completeness of d-separation**
  - For “almost all” distributions where  $P$  factorizes over  $G$ , we have that  $I(G) = I(P)$ 
    - “almost all” distributions: except for a set of measure zero parameterizations of the CPTs (assuming no finite set of parameterizations has positive measure)
    - Means that if  $\mathbf{X}$  and  $\mathbf{Y}$  are **not** d-separated given  $\mathbf{Z}$ , then  $\neg(\mathbf{X} \perp \mathbf{Y} | \mathbf{Z})$

- **Proof sketch for very simple case:**



Polynomials are either identical zero or they are non-zero almost everywhere. Thus, the equality happens with probability 0

d-sep says  $\neg A \perp B$  but it holds  $A \perp B$   
i.e.,  $P(B | A) = P(B)$ , which is a polynomial equality over the space of CPDs

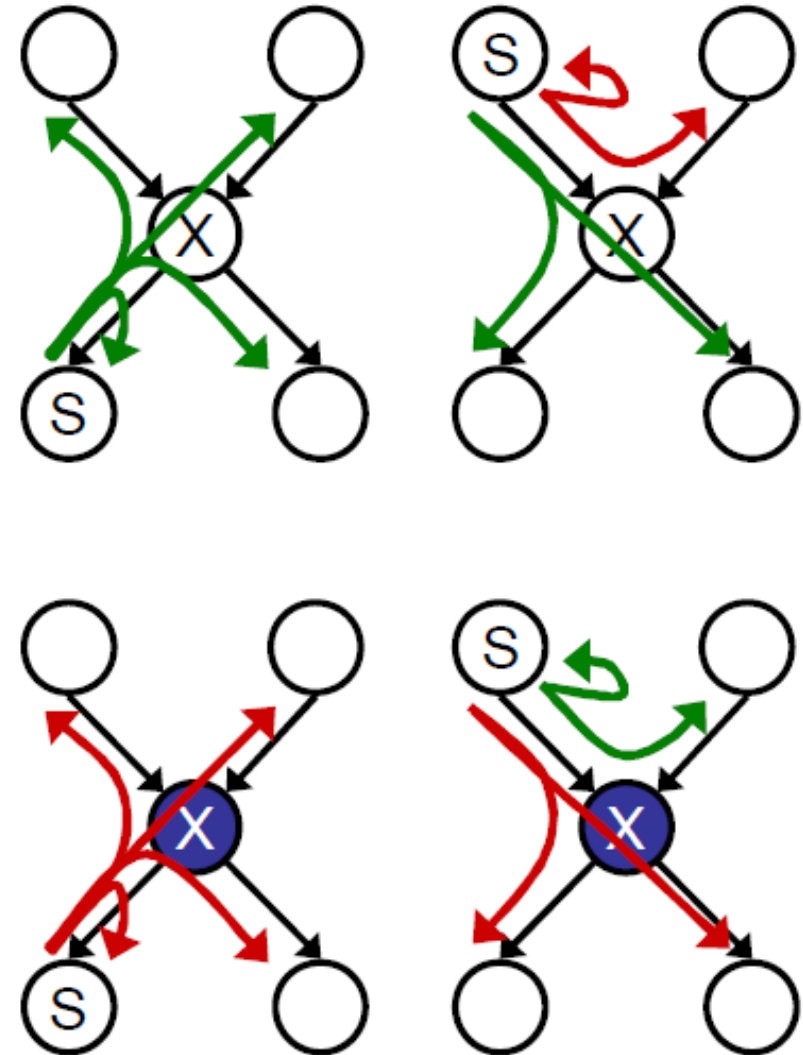
$$\theta_B = \theta_{B|A=t} = \theta_{A=t} \cdot \theta_{B|A=t} + (1 - \theta_{A=t}) \cdot \theta_{B|A=f}$$



# Algorithm for d-separation: The Bayes' Ball



- **Correct algorithm:**
  - Shade in evidence
  - Start at source node
  - Try to reach target by search
- States: pair of (node X, previous state S)
- **Successor function:**
  - **X unobserved:**
    - To any child
    - To any parent if coming from a child
  - **X observed:**
    - From parent to parent
- If you can't reach a node, it's conditionally independent of the start node given evidence

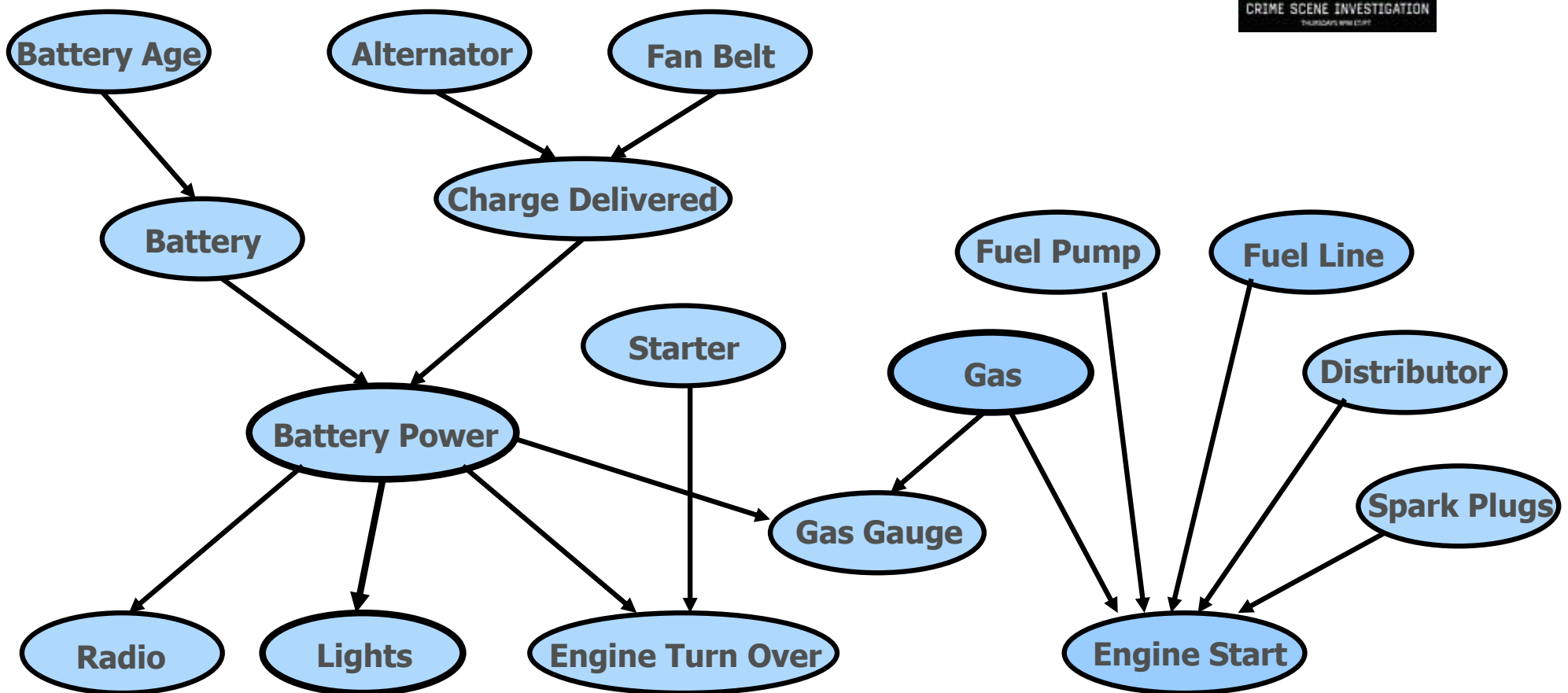


# Interpretation of completeness

- **Theorem: Completeness of d-separation**
  - For “almost all” distributions that  $P$  factorize over  $G$ , we have that  $I(G) = I(P)$
- **BN graph is usually sufficient to capture all independence properties of the distribution!!!!**
- But only for **complete** independence:
  - $P$  says  $(\mathbf{X}=\mathbf{x} \perp \mathbf{Y}=\mathbf{y} \mid \mathbf{Z}=\mathbf{z}), \forall \mathbf{x} \in \text{Val}(\mathbf{X}), \mathbf{y} \in \text{Val}(\mathbf{Y}), \mathbf{z} \in \text{Val}(\mathbf{Z})$
- Often we have **context-specific independence (CSI)**
  - $\exists \mathbf{x} \in \text{Val}(\mathbf{X}), \mathbf{y} \in \text{Val}(\mathbf{Y}), \mathbf{z} \in \text{Val}(\mathbf{Z}): P$  says  $(\mathbf{X}=\mathbf{x} \perp \mathbf{Y}=\mathbf{y} \mid \mathbf{Z}=\mathbf{z})$
  - Many factors may affect your grade



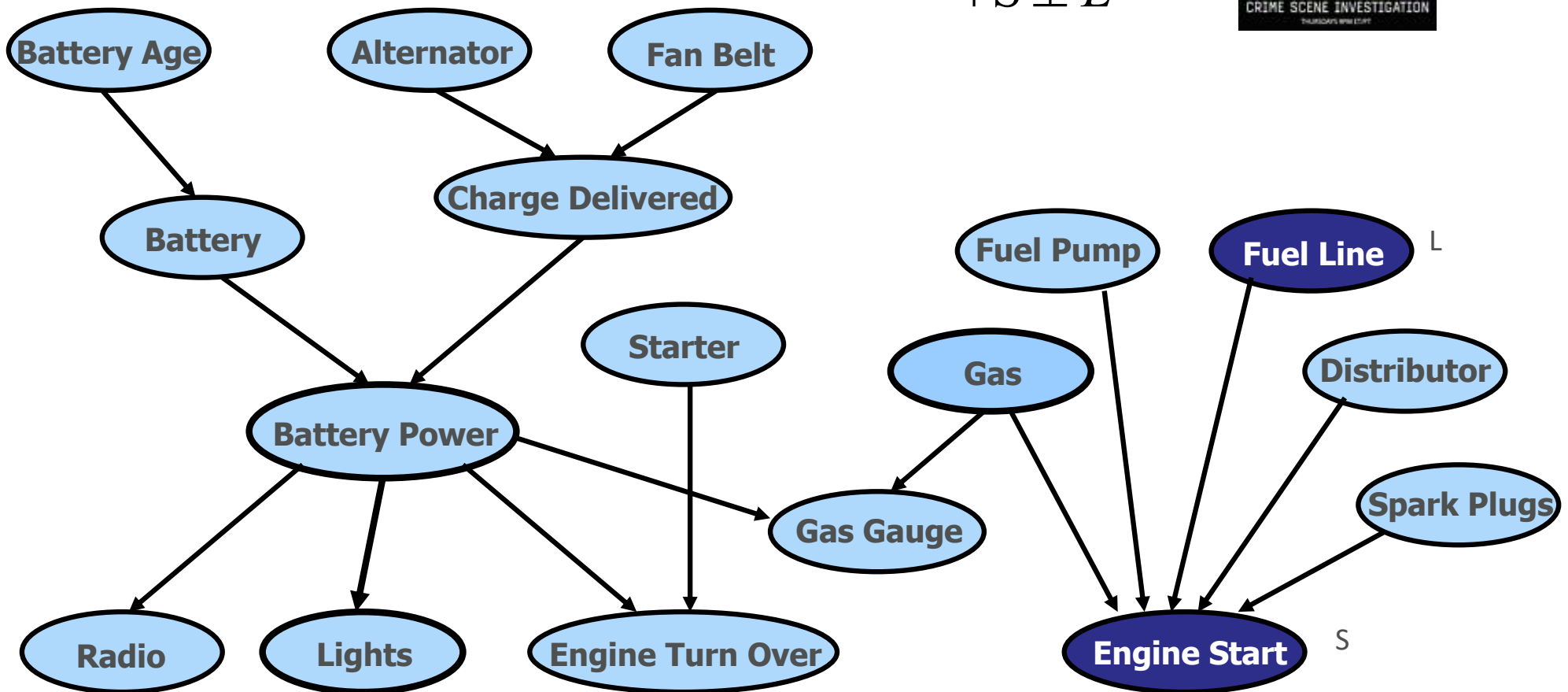
# Excursion: Context specific independence



# Excursion: Context specific independence



$$\neg S \perp L$$



# Excursion: Context specific independence

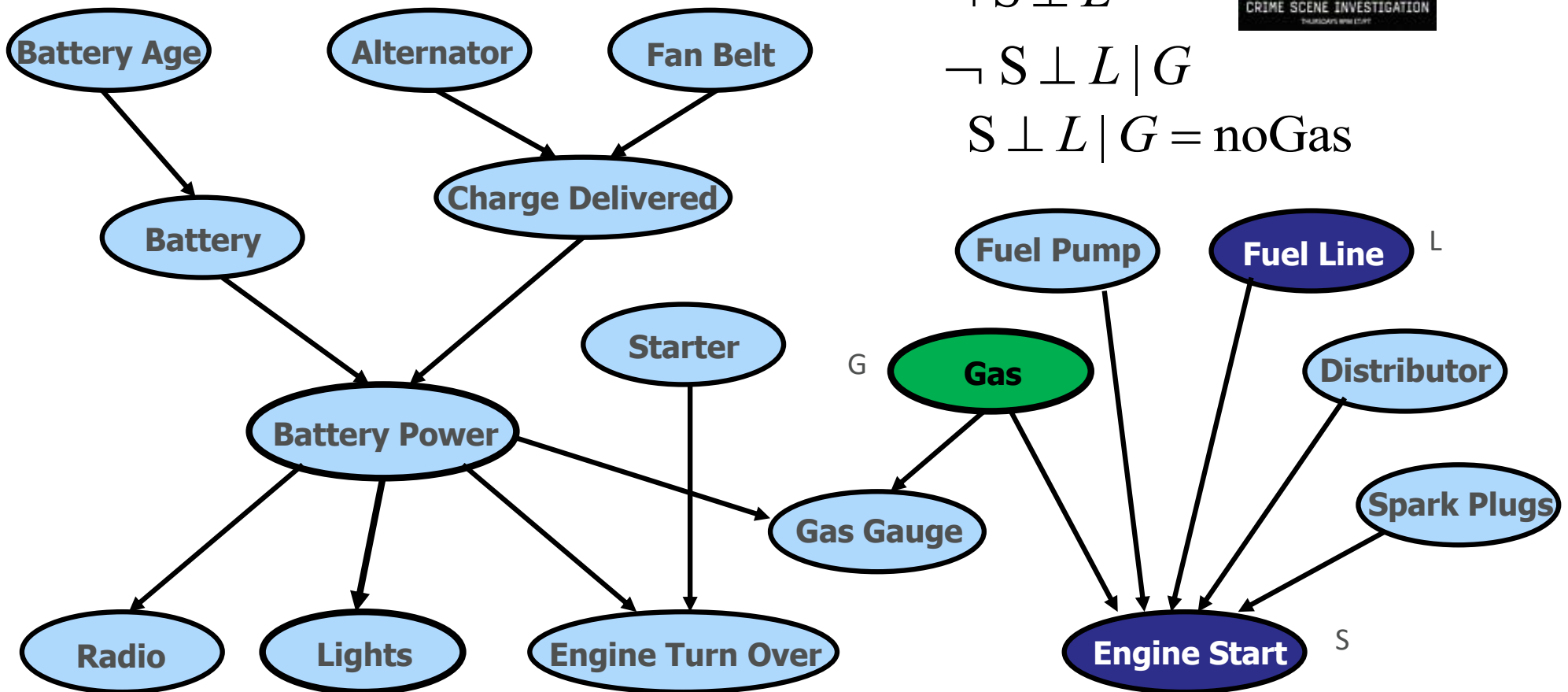
After observing a variable, some vars become independent



$$\neg S \perp L$$

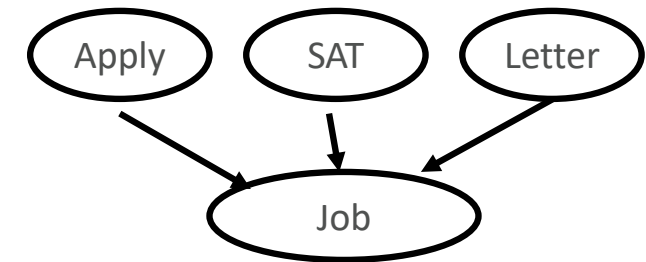
$$\neg S \perp L | G$$

$$S \perp L | G = \text{noGas}$$



# CSI Representation: Tree CPD

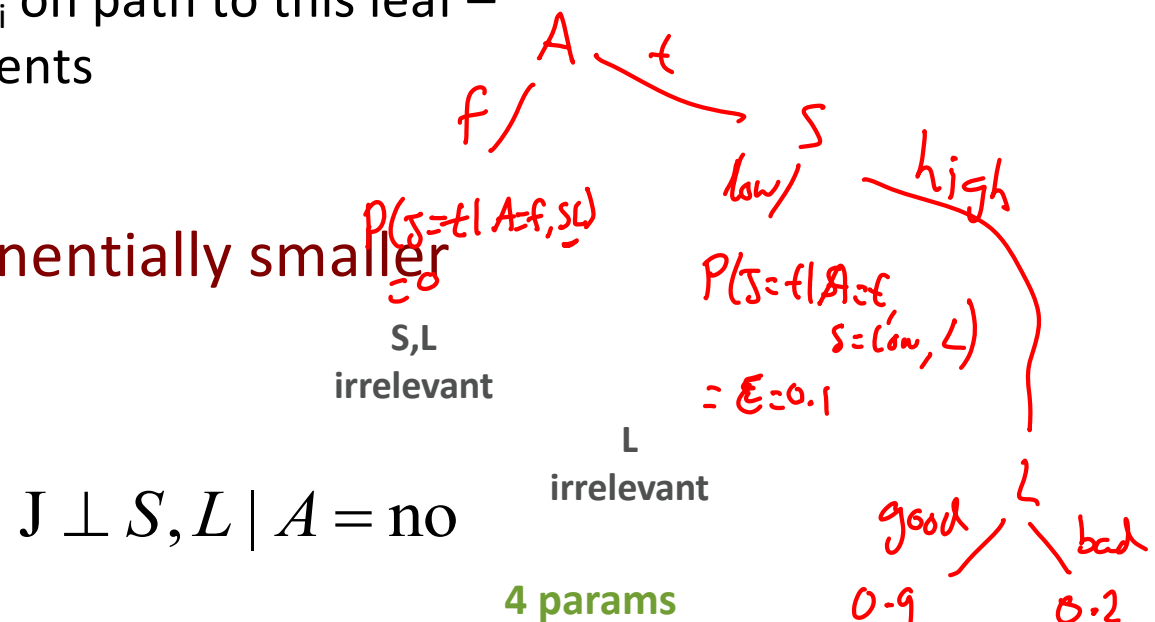
- Represent  $P(X_i | \mathbf{Pa}_{X_i})$  using a decision tree
  - Path to leaf is an assignment to (a subset of)  $\mathbf{Pa}_{X_i}$
  - Leaves are distributions over  $X_i$  given assignment of  $\mathbf{Pa}_{X_i}$  on path to leaf
- **Interpretation of leaf:**
  - For specific assignment of  $\mathbf{Pa}_{X_i}$  on path to this leaf –  $X_i$  is independent of other parents
- Representation can be exponentially smaller than equivalent table



Assuming binary RVs

Table:  $O(2^k)$

In the structure:  $O(k)$



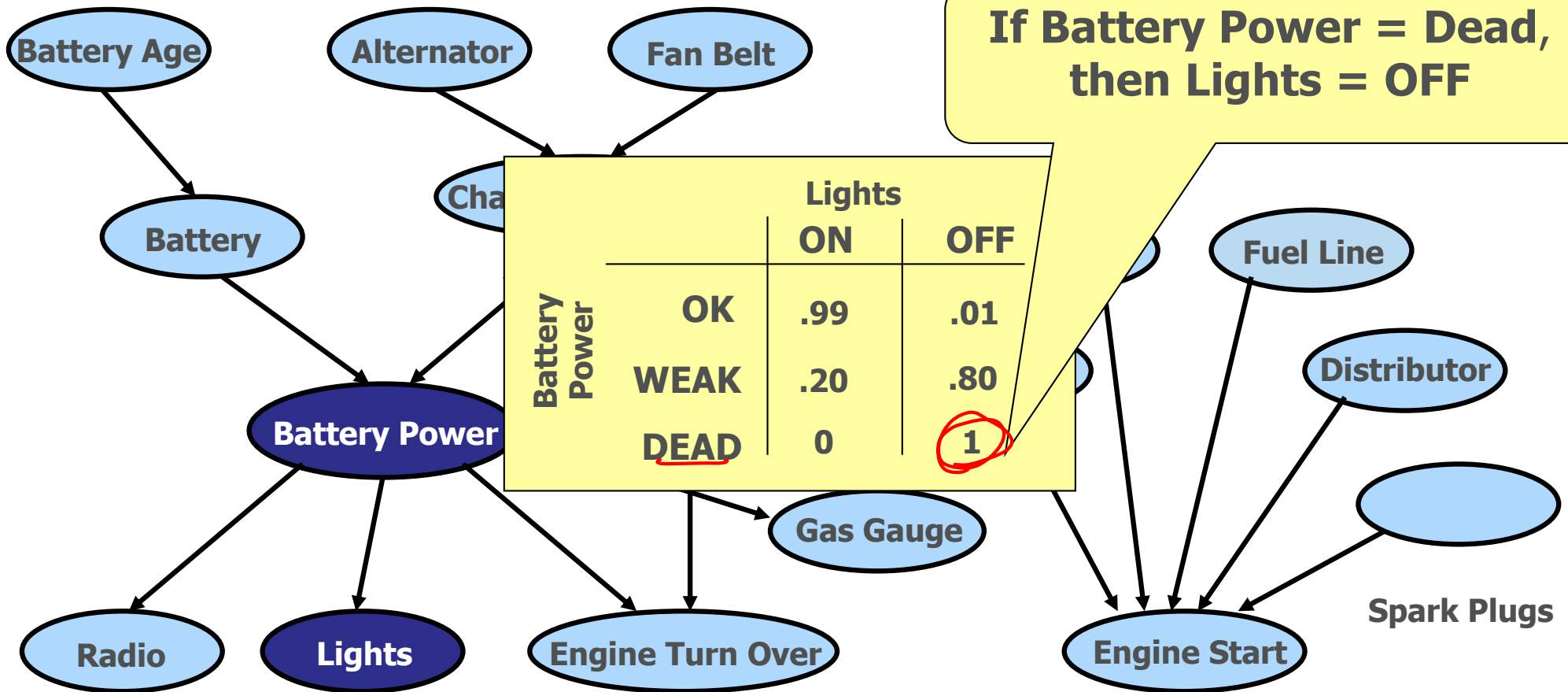




# Other CSIs: Determinism

Determinism

If Battery Power = Dead,  
then Lights = OFF



## Determinism and inference

- Determinism gives a little sparsity in table, **but much bigger impact on inference**
- Multiplying deterministic factor with other factor often **introduces many new zeros**
  - Operations related to theorem proving, e.g., unit resolution

		Lights	
		ON	OFF
Battery Power	OK	.99	.01
	WEAK	.20	.80
	DEAD	0	1

$g(Y) = \prod f_j$  and  $f_i(x_i) = 0$  then  
all values of the  $g(Y)$  table where  
 $X_i = x_i$  will be zero

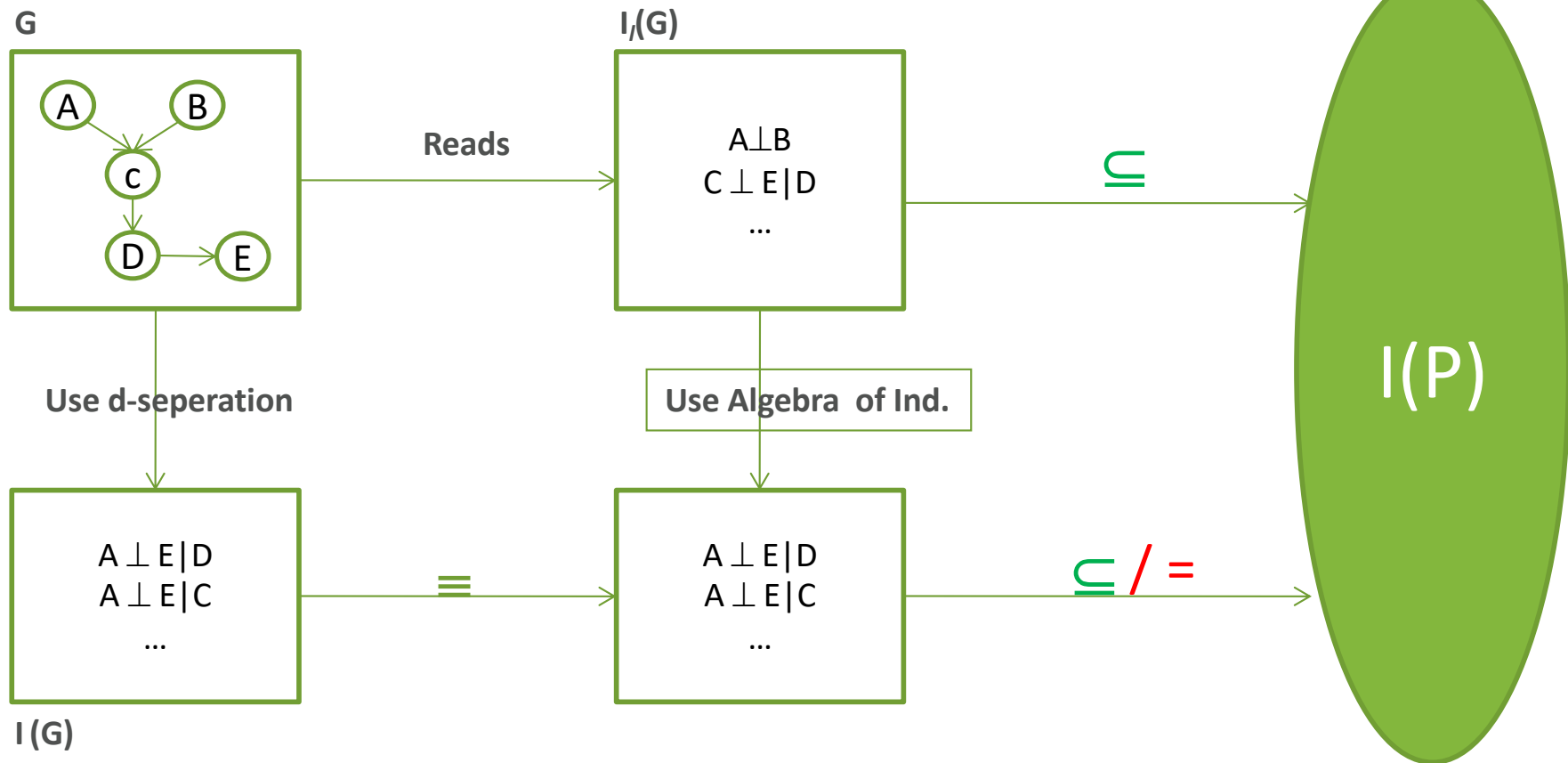


# State-of-the-art Models ...

- **Often characterized by:**
  - Richness in local structure (determinism, CSI)
  - Massiveness in size (10,000's variables)
  - High connectivity (treewidth, explained later in class)
- **Enabled by:**
  - High level modeling tools:
    - **relational, first order (more about this maybe later in class!!!!)**
  - Advances in machine learning
  - New application areas (synthesis):
    - Bioinformatics (e.g. linkage analysis)
    - Sensor networks
- **Exploiting local and relational structure a must!**



# What you need to know



- If  $G$  is an I-map of  $P$  then  $I_l(G) \subseteq I(P)$
- Also, it is always true that  $I(G) \subseteq I(P)$  means d-separation is **sound**
- And for almost all  $P$ s that factor over  $G$ ,  $I(G) = I(P)$ ,  $P$  is **faithful** to  $G$



# What you need to know

- d-separation and independence
  - sound procedure for finding independencies  $I(G) \subseteq I(P)$
  - existence of distributions with these independencies
  - (almost) all independencies can be read directly from graph without looking at CPTs  $I(G) = I(P)$
- Context-specific independence (CSI)
- Bayes' Ball



# What's next

- Inference



# Is Inference in BNs hopeless?

- In general, yes!
  - Even approximate!
- In practice
  - Exploit structure
  - Many effective approximation algorithms (some with guarantees)
- For now, we'll talk about exact inference
  - Approximate inference later this semester

## Theorem:

Inference in Bayesian networks  
(even approximate, without  
proof) is NP-hard



## General probabilistic inference

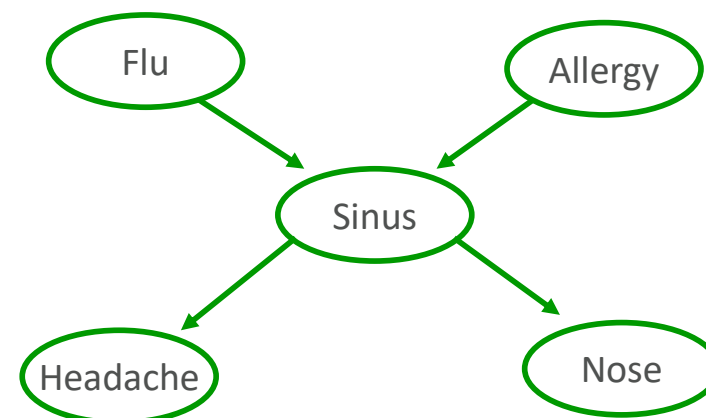
- Query:  $P(X \mid e)$

- Using def. of cond. prob.:

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

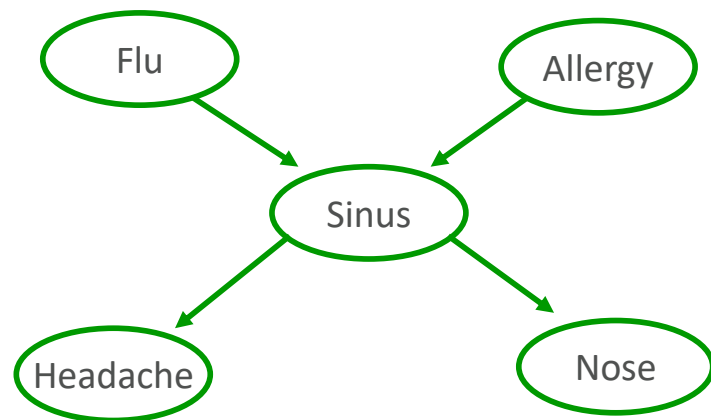
- Normalization: (we stopped here)

$$P(X \mid e) \propto P(X, e)$$





# Probabilistic inference example



$$P(A|N=t) \propto P(A, N=t)$$

$$= \sum_f \sum_s \sum_h P(A, f, s, h, N=t)$$

$$= \sum_f \sum_s \sum_h P(f) P(A) P(s|f, A) P(h|s) P(N=t|s)$$

8 terms

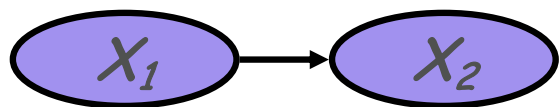
4 multiplies

$2^3 \leftarrow \# \text{ eliminated vars}$  32 multiplies

Inference seems exponential in number of variables!



# Inference in Simple Chains



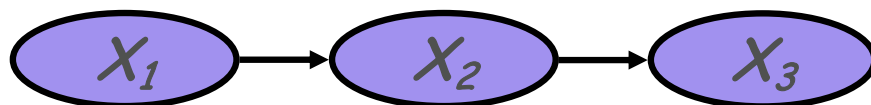
How do we compute

$$P(x_2)$$

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} \underbrace{P(x_1)}_{\text{CPDs}} \underbrace{P(x_2 | x_1)}_{\text{CPDs}}$$



## Inference in Simple Chains (cont.)



How do we compute  $P(x_3)$ ?

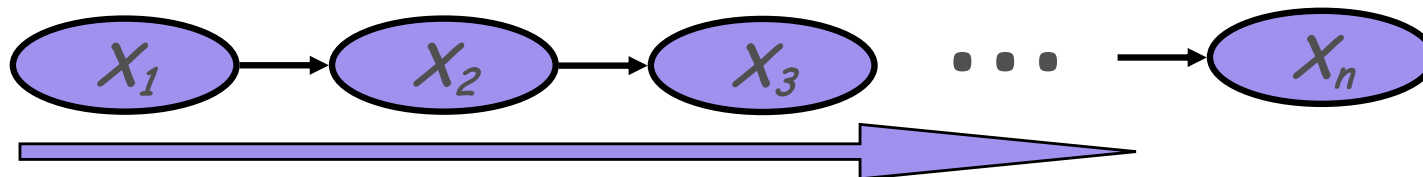
$$P(x_3) = \sum_{x_2} P(x_2, x_3) = \sum_{x_2} \underbrace{P(x_2)}_{\text{computed}} \underbrace{P(x_3 | x_2)}_{\text{CPD}}$$

- We already know how to compute  $P(x_2)$  ...

$$P(x_2) = \sum_{x_1} P(x_1, x_2) = \sum_{x_1} P(x_1) P(x_2 | x_1)$$



## Inference in Simple Chains (cont.)



How do we compute

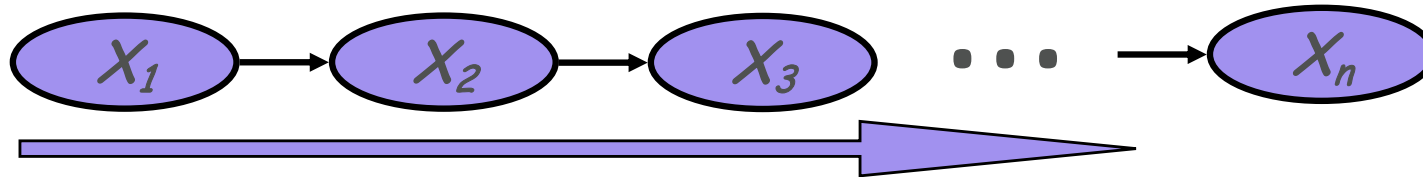
$$P(x_n)$$

- Iteratively compute  $P(x_1), P(x_2), P(x_3), \dots$  using

$$P(x_{i+1}) = \sum_{\substack{x_i \text{ computed}}} \underbrace{P(x_i)}_{\text{CPD}} \underbrace{P(x_{i+1} | x_i)}_{\text{CPD}}$$



# Complexity of inference: Simple Chains



$O(n \cdot k^2)$  v.s. exponentially in  $n$

# Variable Elimination

General idea:

- Write query in the form

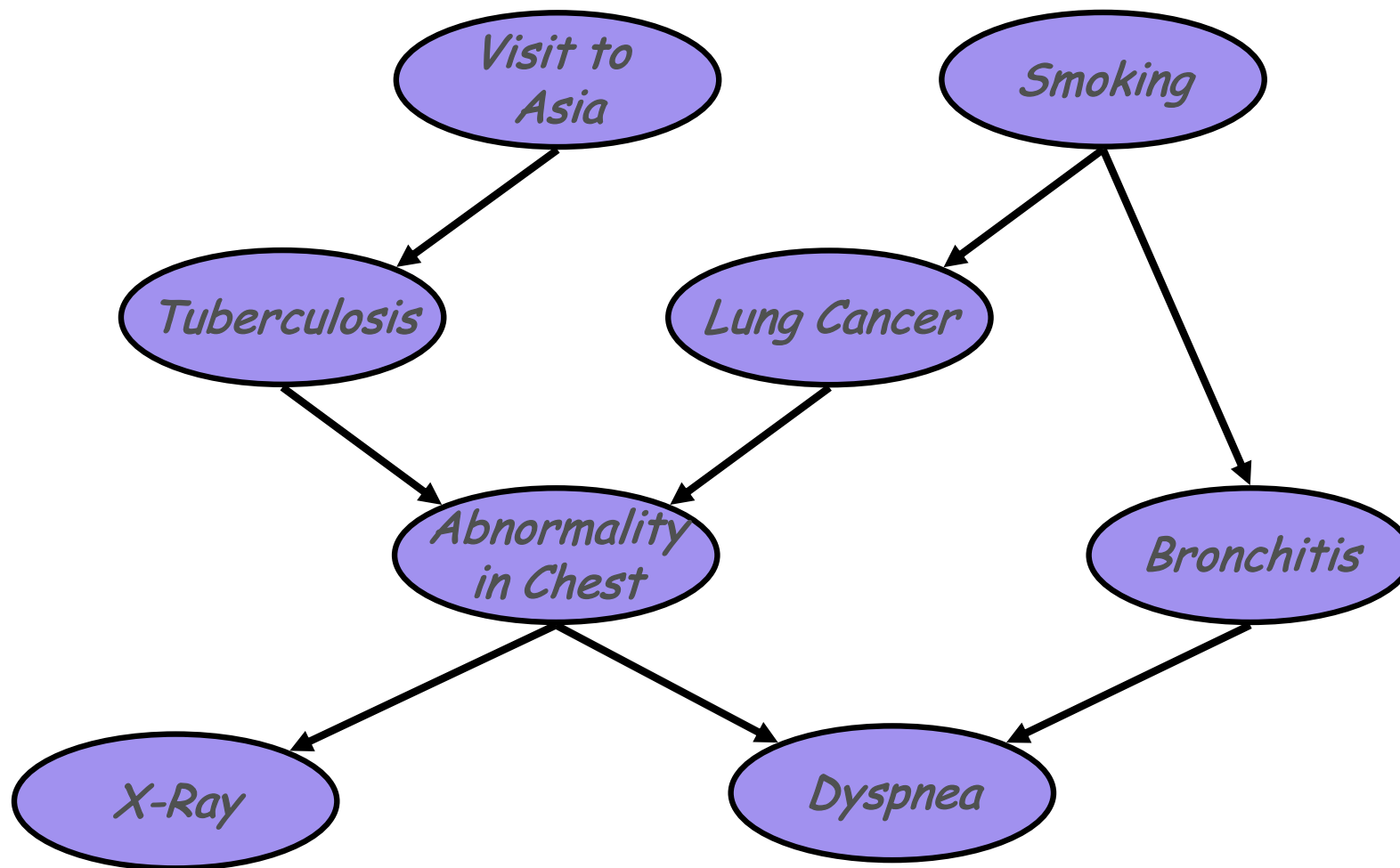
$$P(x_n, \mathbf{e}) = \sum_{x_k} \cdots \sum_{x_3} \sum_{x_2} \prod_i P(x_i | pa_i)$$

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



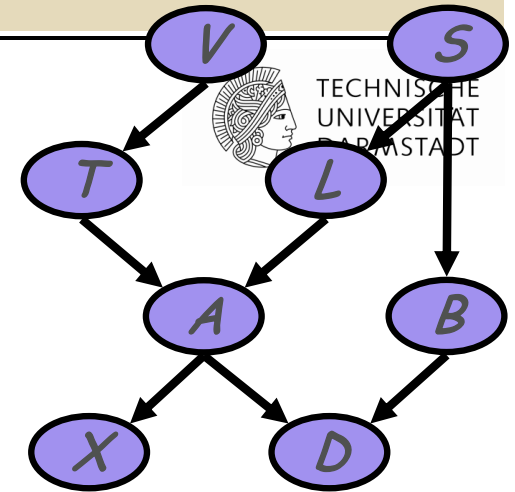
# A More Complex Example

- “Asia” network:



- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



$$\underline{P(v)} \underline{P(s)} P(t | v) P(l | s) P(b | s) P(a | t, l) P(x | a) P(d | a, b)$$

Eliminate:  $v$

$$\text{Compute: } f_v(t) = \sum_v P(v) P(t | v)$$

$$\Rightarrow \underline{f_v(t)} P(s) P(l | s) P(b | s) P(a | t, l) P(x | a) P(d | a, b)$$

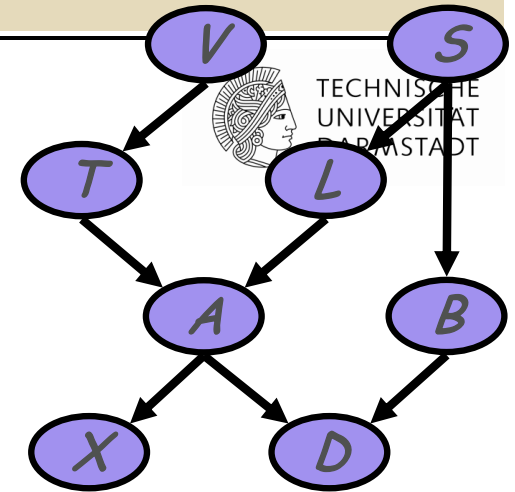
Note:  $f_v(t) = P(t)$

In general, result of elimination is not necessarily a probability term



- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



$$P(v)P(s)P(t | v)P(l | s)P(b | s)P(a | t, l)P(x | a)P(d | a, b)$$

$$\Rightarrow f_v(t) \underline{P(s)} \underline{P(l | s)} \underline{P(b | s)} P(a | t, l) P(x | a) P(d | a, b)$$

Eliminate:  $s$

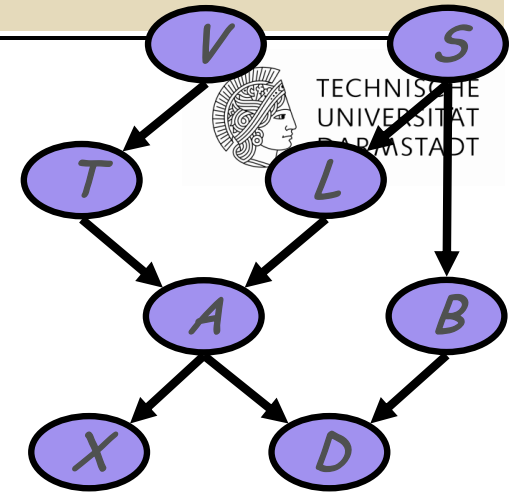
$$\text{Compute: } f_s(b, l) = \sum_s P(s) P(b | s) P(l | s)$$

$$\Rightarrow f_v(t) \underline{f_s(b, l)} P(a | t, l) P(x | a) P(d | a, b)$$

Summing on  $s$  results in a factor with two arguments  $f_s(b, l)$   
 In general, result of elimination may be a function of several variables

- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



$$\begin{aligned}
 &P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow &f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow &f_v(t)f_s(b,l)\underline{P(a|t,l)}P(x|a)P(d|a,b)
 \end{aligned}$$

Eliminate:  $x$

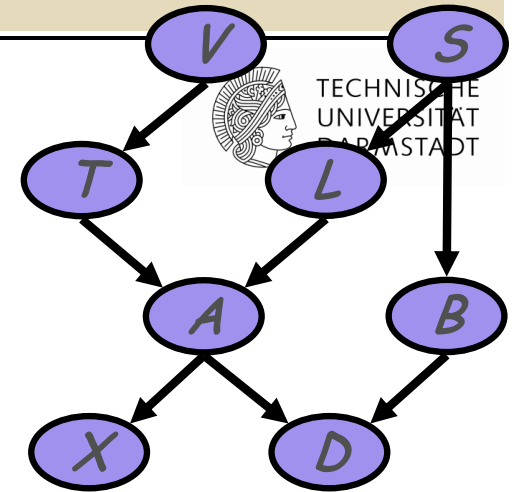
Compute:  $f_x(a) = \sum_x P(x|a)$

$$\Rightarrow f_v(t)f_s(b,l)\underline{f_x(a)}P(a|t,l)P(d|a,b)$$

Note:  $f_x(a) = 1$  for all values of  $a$  !!

- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



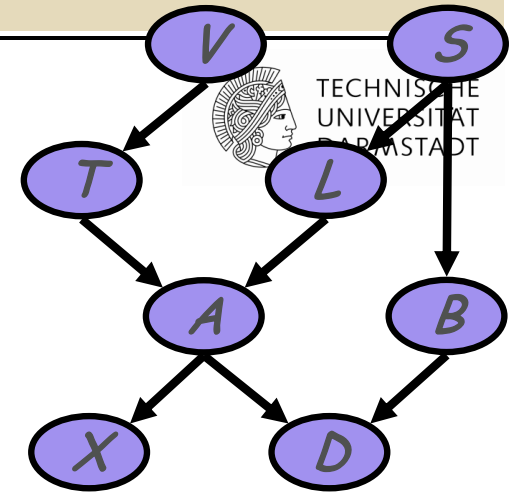
$$\begin{aligned}
 & P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & \underline{f_v(t)}f_s(b,l)f_x(a)\underline{P(a|t,l)}P(d|a,b)
 \end{aligned}$$

Eliminate:  $t$

$$\begin{aligned}
 \text{Compute: } & f_t(a,l) = \sum_t f_v(t)P(a|t,l) \\
 \Rightarrow & f_s(b,l)f_x(a)\underline{f_t(a,l)}P(d|a,b)
 \end{aligned}$$

- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



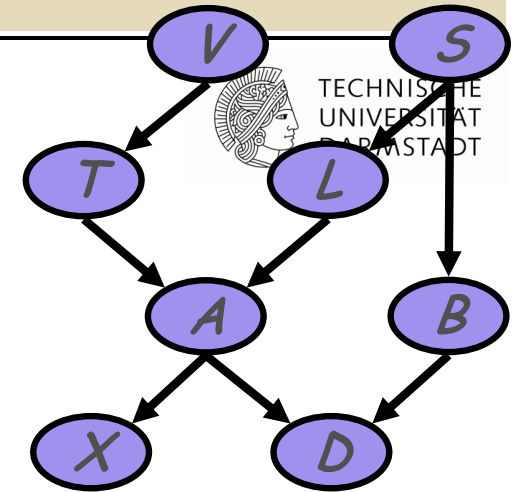
$$\begin{aligned}
 & P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b) \\
 \Rightarrow & \underline{f_s(b,l)}\underline{f_x(a)}\underline{f_t(a,l)}P(d|a,b)
 \end{aligned}$$

Eliminate: /

$$\begin{aligned}
 \text{Compute: } & f_l(a,b) = \sum f_s(b,l)f_t(a,l) \\
 \Rightarrow & \underline{f_l(a,b)}f_x(a)P^l(d|a,b)
 \end{aligned}$$

- We want to compute  $P(d)$
- Need to eliminate:  $v, s, x, t, l, a, b$

Initial factors



$$\begin{aligned}
 & P(v)P(s)P(t|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)P(s)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)f_s(b,l)P(a|t,l)P(x|a)P(d|a,b) \\
 \Rightarrow & f_v(t)f_s(b,l)f_x(a)P(a|t,l)P(d|a,b) \\
 \Rightarrow & f_s(b,l)f_x(a)f_t(a,l)P(d|a,b) \\
 \Rightarrow & \underline{f_l(a,b)}\underline{f_x(a)}\underline{P(d|a,b)} \Rightarrow \underline{f_a(b,d)} \Rightarrow \underline{f_b(d)}
 \end{aligned}$$

Eliminate:  $a, b$

$$\text{Compute: } f_a(b,d) = \sum_a f_l(a,b)f_x(a)p(d|a,b) \quad f_b(d) = \sum_b f_a(b,d)$$

# Variable Elimination

- We now understand variable elimination as a sequence of **rewriting** operations
- Computation depends on order of elimination



## Dealing with Evidence

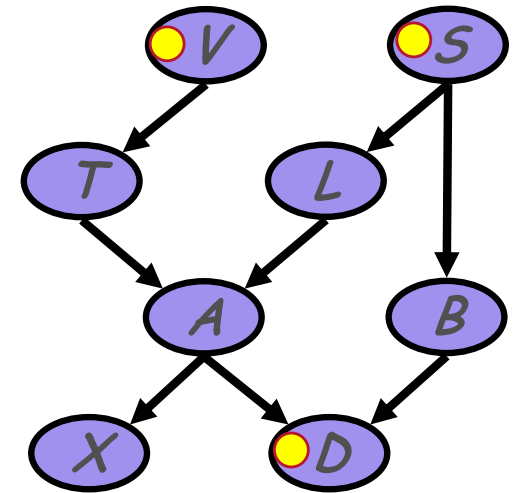
- How do we deal with evidence?

- Suppose get evidence

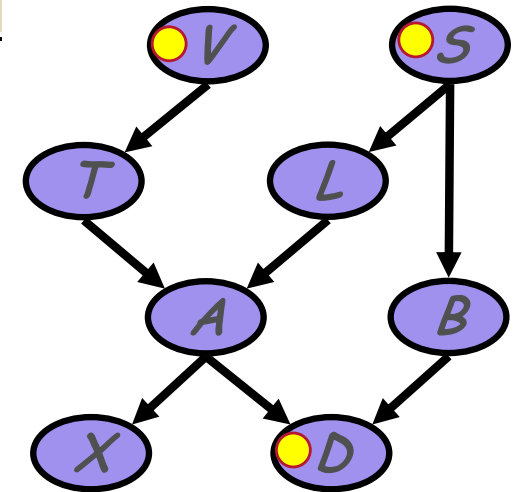
$$V = t, S = f, D = t$$

- We want to compute

$$P(L, V = t, S = f, D = t)$$



## Dealing with Evidence



- We start by writing the factors:

$$P(v)P(s)P(t | v)P(l | s)P(b | s)P(a | t, l)P(x | a)P(d | a, b)$$

- Since we know that  $V = t$ , we don't need to eliminate  $V$
- Instead, we can replace the factors  $P(V)$  and  $P(T|V)$  with

$$f_{P(V)} = P(V = t) \quad f_{P(T|V)}(T) = P(T | V = t)$$

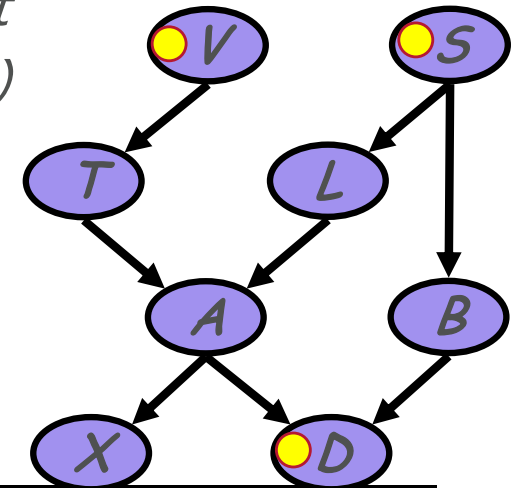
- This “selects” the appropriate parts of the original factors given the evidence
- Note that  $f_{P(V)}$  is a constant, and thus does not appear in elimination of other variables





- Given evidence  $V = t, S = f, D = t$
- Compute  $P(L, V = t, S = f, D = t)$

## Dealing with Evidence

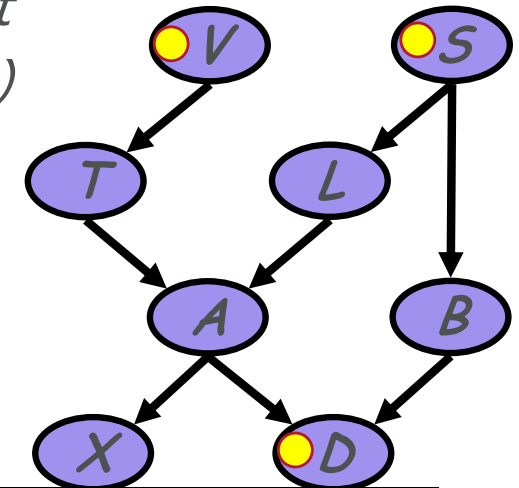


- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) P(x | a) f_{P(d|a,b)}(a, b)$$

- Given evidence  $V = t, S = f, D = t$
- Compute  $P(L, V = t, S = f, D = t)$

## Dealing with Evidence



- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) \underline{P(a | t, l)} P(x | a) f_{P(d|a,b)}(a, b)$$

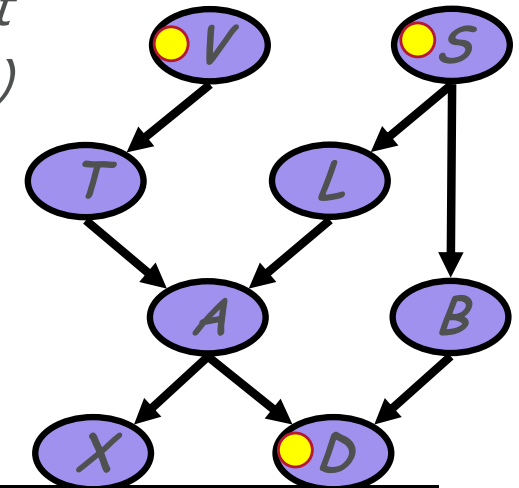
- Eliminating  $X$ , we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) f_x(a) f_{P(d|a,b)}(a, b)$$



- Given evidence  $V = t, S = f, D = t$
- Compute  $P(L, V = t, S = f, D = t)$

## Dealing with Evidence



- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) \underline{P(a|t,l)} P(x|a) f_{P(d|a,b)}(a,b)$$

- Eliminating  $x$ , we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) \underline{P(a|t,l)} f_x(a) f_{P(d|a,b)}(a,b)$$

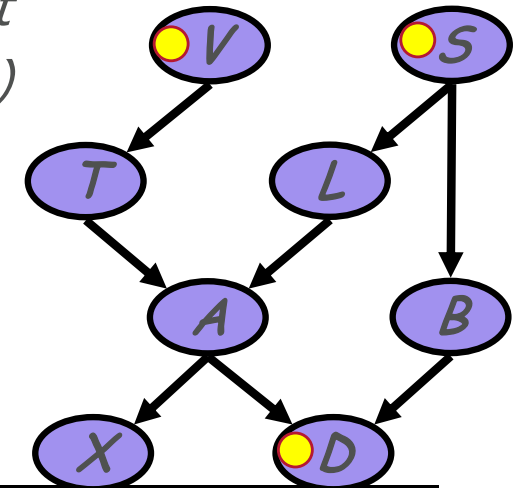
- Eliminating  $t$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) \underline{f_t(a,l)} f_x(a) f_{P(d|a,b)}(a,b)$$



- Given evidence  $V = t, S = f, D = t$
- Compute  $P(L, V = t, S = f, D = t)$

## Dealing with Evidence



- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) P(x | a) f_{P(d|a,b)}(a, b)$$

- Eliminating  $x$ , we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) f_x(a) f_{P(d|a,b)}(a, b)$$

- Eliminating  $t$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_t(a, l) f_x(a) f_{P(d|a,b)}(a, b)$$

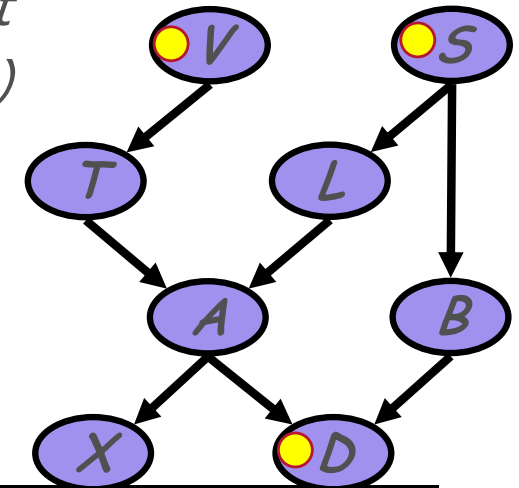
- Eliminating  $a$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_a(b, l)$$



- Given evidence  $V = t, S = f, D = t$
- Compute  $P(L, V = t, S = f, D = t)$

## Dealing with Evidence



- Initial factors, after setting evidence:

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) P(x | a) f_{P(d|a,b)}(a, b)$$

- Eliminating  $x$ , we get

$$f_{P(v)} f_{P(s)} f_{P(t|v)}(t) f_{P(l|s)}(l) f_{P(b|s)}(b) P(a | t, l) f_x(a) f_{P(d|a,b)}(a, b)$$

- Eliminating  $t$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_t(a, l) f_x(a) f_{P(d|a,b)}(a, b)$$

- Eliminating  $a$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_{P(b|s)}(b) f_a(b, l)$$

- Eliminating  $b$ , we get

$$f_{P(v)} f_{P(s)} f_{P(l|s)}(l) f_b(l)$$



# Summary: Variable elimination algorithm

- Given a BN and a query  $P(X|e) / P(X,e)$
- Instantiate evidence  $e$
- Prune non-active vars for  $\{X,e\}$
- Choose an ordering on variables, e.g.,  $X_1, \dots, X_n$
- Initial *factors*  $\{f_1, \dots, f_n\}$ :  $f_i = P(X_i | \mathbf{Pa}_{X_i})$  (CPT for  $X_i$ )
- For  $i = 1$  to  $n$ , If  $X_i \notin \{X, E\}$  *← must be eliminated*
  - Collect factors  $f_1, \dots, f_k$  that include  $X_i$
  - Generate a new factor by eliminating  $X_i$  from these factors

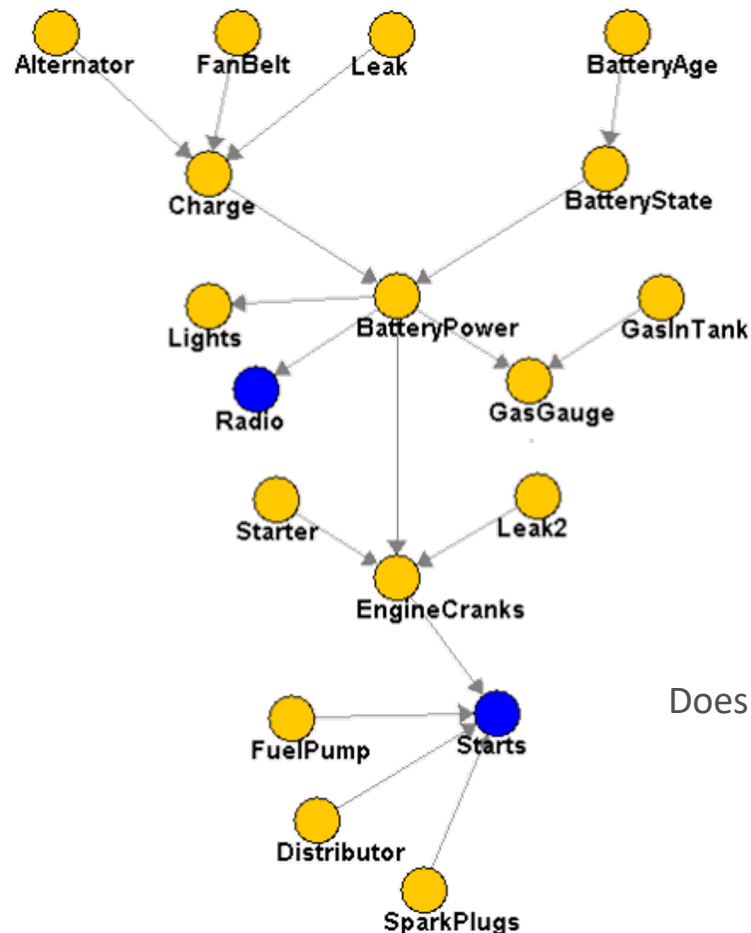
IMPORTANT!!!

$$g = \sum_{X_i} \prod_{j=1}^k f_j$$

- Variable  $X_i$  has been eliminated! Add  $g$  to the set of factors
- Normalize  $P(X,e)$  to obtain  $P(X|e)$



# Complexity of VE: (Poly)-tree graphs



Variable elimination order:  
Start from “leaves” inwards:

- Start from skeleton!
- Choose a “root”, any node
- Find topological order for root
- Eliminate variables in reverse order

Does not create factors any bigger than original CPTs

**Linear in CPT sizes!!! (versus exponential)**



# Complexity of VE: General Case

- During VE, we **multiply** and marginalize factors

$$g = \sum_{X_i} \prod_{j=1}^k f_j$$

- **Mult.:** exponential in the number of involved vars
- **Marg. :** linear in the size of the product factor
- **Exponential in the # of vars in intermediate factors**, i.e., dominated by the largest intermediate factor





## What's next

- Thus far: Variable elimination
  - (Often) Efficient algorithm for inference in graphical models
- Next: Understanding complexity of variable elimination in more detail
  - Will lead to cool junction tree algorithm later

