Donders Institute
for Brain, Cognition and Behaviour

BKI259: Artificial Intelligence: Principles and Techniques

Bayesian Networks (part 2/3)

Radboud University Nijmegen

Lecture outline

- Today we discuss **inference** in Bayesian networks: computing $P(X \mid Y)$ for any $X, Y \subseteq V$
- Short recap of probability axioms to allow for ‘naive’ inference computations
- Factors and the **Variable elimination** algorithm
- Computational **complexity** of Inference (sketch)

Axioms of probability theory

- P is a *probability measure* over a set Ω
- P should obey three axioms:
  1. $P(A) \geq 0$ for all events A
  2. $P(\Omega) = 1$
  3. $P(A \cup B) = P(A) + P(B)$ for disjoint events A and B
- Some consequences:
  - $P(A) = 1 - P(\Omega \setminus A)$
  - If $A \subseteq B$ then $P(A) \leq P(B)$
  - $P(A \cup B) = P(A) + P(B) - P(A \cap B) \leq P(A) + P(B)$

- Given these axioms and a completely defined probability measure any quantity of interest can be computed!

Useful formulas

- Probabilities over all disjoint subsets sum to 1:

$$\sum_{x \in X} P(X = x) = 1$$

- $P(x)$ shorthand notation for $P(X=x)$

- So, if the variable A has three possible values: $A = \{a_1 , a_2, a_3\}$, thus $A=a_1$ or $A=a_2$ or $A=a_3$, then $P(a_1) + P(a_2) + P(a_3) = 1$

- This also holds for conditional probabilities: $P(a_1|e) + P(a_2|e) + P(a_3|e) = 1$

Important notions

- Joint distribution $P(X_1, X_2, \ldots, X_n)$
  - Example: $P(A, B)$, $A \in \{a_1, a_2, a_3\}$, $B \in \{b_1, b_2\}$

A \ B	b_1	b_2	$P(A)$
a_1	0.05	0.54	0.59
a_2	0.08	0.04	0.12
a_3	0.17	0.12	0.29
$P(B)$	0.30	0.70	

$P(A) = P(A,b_1)+P(A,b_2)$

A	**B**	***P(A,B)***
a_1	b_1	0.05
a_1	b_2	0.54
a_2	b_1	0.08
a_2	b_2	0.04
a_3	b_1	0.17
a_3	b_2	0.12

  - Marginal distribution $P(X_i)$:

$$P(X_i) = \sum_{x_1 \in X_1} \sum_{x_2 \in X_2} \ldots \sum_{x_{i-1} \in X_{i-1}} \sum_{x_{i+1} \in X_{i+1}} \ldots \sum_{x_n \in X_n} P(X_1 = x_1, X_2 = x_2, \cdots, X_i, \ldots, X_n = x_n)$$

Remember

- Conditional probability: $P(H \mid E) = P(H,E) / P(E)$
- Bayes rule: $P(H \mid E) = P(H) \times P(E \mid H)$ / P(E)
- Bayesian network: Directed acyclic graph, edges from parents to children, the parents of a node X_i come before X_i in the ordering (X_1, X_2, ... ,X_i, ... ,X_n)
- The joint probability distribution for a Bayesian network can be *factorized*:

Chain or product rule

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, \ldots, X_{i-1}) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$$

BN property

where Parents(X_i) are the parents of X_i in the graph

Inference in BNs

- Example:

P(tampering) = 0.02

P(fire) = 0.01

P(alarm | fire $\wedge$ tampering) = 0.5

P(alarm | fire $\wedge$ $\neg$tampering) = 0.99

P(alarm | $\neg$fire $\wedge$ tampering) = 0.85

P(alarm | $\neg$fire $\wedge$ $\neg$tampering) = 0.0001

P(smoke | fire) = 0.9

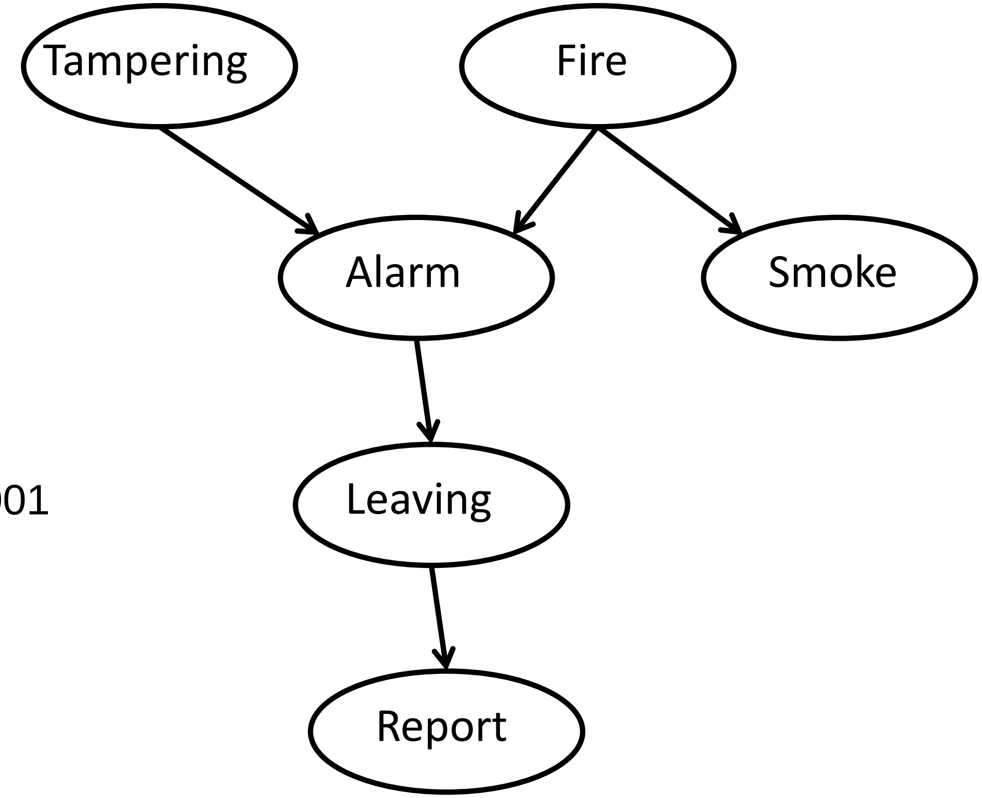
P(smoke | $\neg$fire) = 0.01

P(leaving | alarm) = 0.88

P(leaving | $\neg$alarm) = 0.001

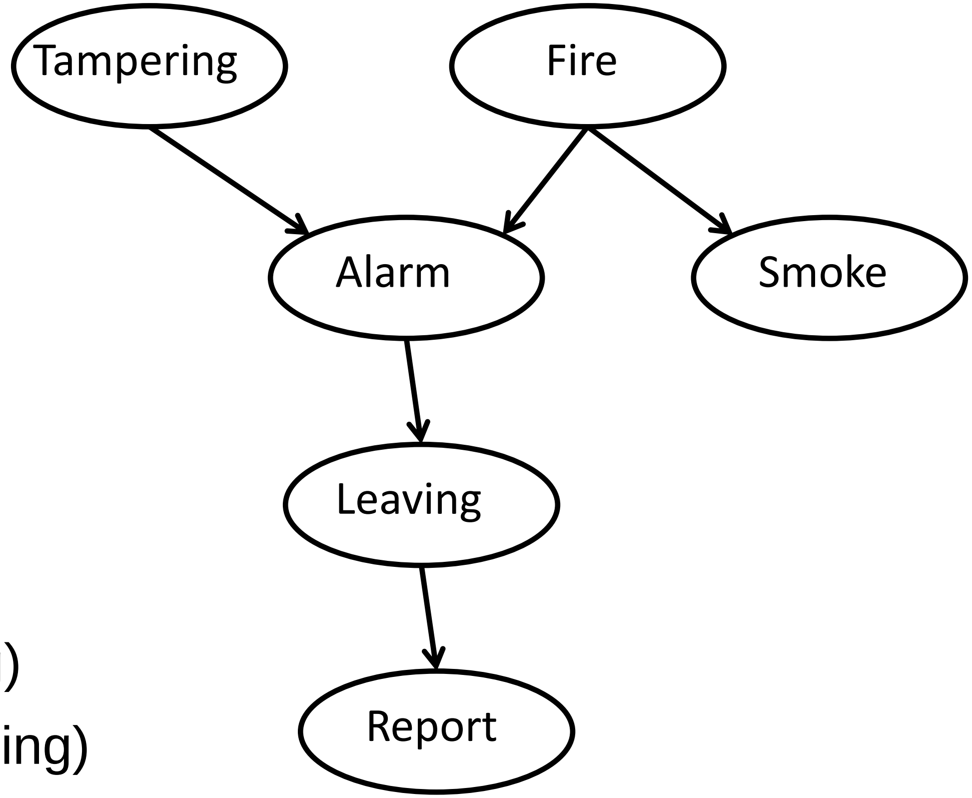
P(report | leaving) = 0.75

P(report | $\neg$leaving) = 0.01



Inference in BNs

- Compute:
  - P(Alarm)
  - P(Report)
  - P(Alarm | Tampering)
  - P(Smoke | Alarm)
  - P(Report | Fire)
  - P(Fire | Report)
  - P(Fire, Alarm | Leaving)
  - P(Fire | Alarm, Tampering)
  - ...



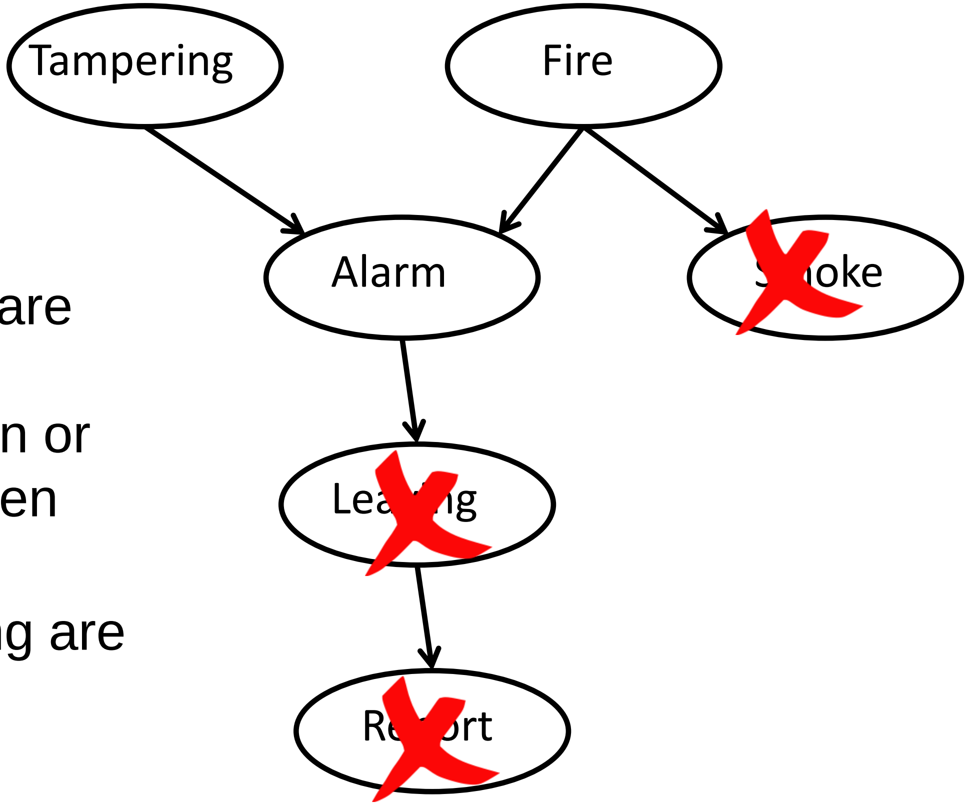
Inference in BNs

- Compute:
  - P(Alarm)

- **Barren nodes**: nodes that are neither evidence nor target nodes, and have no children or only barren nodes as children

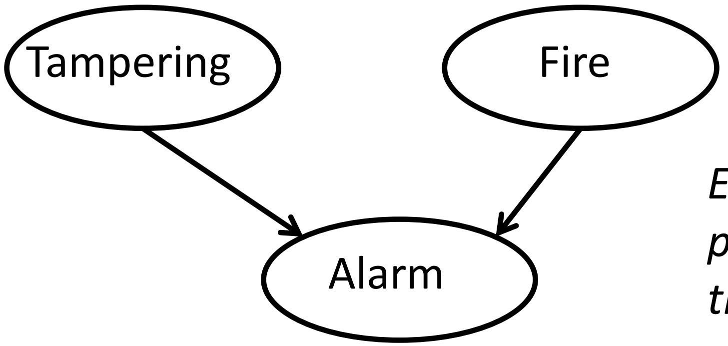
- Smoke, Report, and Leaving are barren nodes

- Barren nodes can be discarded for this particular computation



Inference in BNs

- Compute:
  - P(Alarm)



Essential part of the BN

$$P(Alarm) = \sum_{Tampering, Fire} P(Alarm, Tampering, Fire) \overset{BN}{=}$$

$$\sum_{Tampering, Fire} P(Alarm \mid Tampering, Fire) P(Tampering) P(Fire) =$$

$$P(Alarm \mid tamp, fire) P(tamp) P(fire) +$$

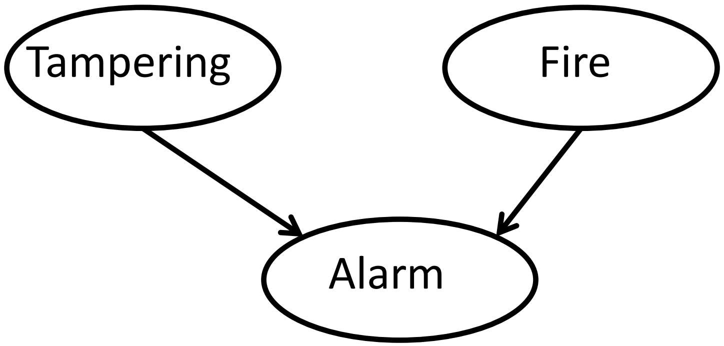
$$P(Alarm \mid tamp, \neg fire) P(tamp) P(\neg fire) +$$

$$P(Alarm \mid \neg tamp, fire) P(\neg tamp) P(fire) +$$

$$P(Alarm \mid \neg tamp, \neg fire) P(\neg tamp) P(\neg fire) =$$

Inference in BNs

- Compute: P(Alarm)



Alarm	P(Alarm)
alarm	0.0266
¬alarm	0.9734

$$
\begin{aligned}
&P(Alarm) = \\
&P(Alarm \mid tamp, fire)P(tamp)P(fire) + \\
&P(Alarm \mid tamp, \neg fire)P(tamp)P(\neg fire) + \\
&P(Alarm \mid \neg tamp, fire)P(\neg tamp)P(fire) + \\
&P(Alarm \mid \neg tamp, \neg fire)P(\neg tamp)P(\neg fire) = \\
&\begin{pmatrix} 0.05*0.02*0.01+0.85*0.02*0.99+0.99*0.98*0.01+0.0001*0.98*0.99 \\ 0.95*0.02*0.01+0.15*0.02*0.99+0.01*0.98*0.01+0.9999*0.98*0.99 \end{pmatrix} = \\
&\begin{pmatrix} 0.0266 \\ 0.9734 \end{pmatrix}
\end{aligned}
$$

Can we do computations smarter?

- This computation can blow up quickly – can we do any faster? Yes, in several distinct ways!

We look at nodes as objects that can pass around information to connected nodes. Keeps computations local, works only on trees.

Judea Pearl's message passing algorithm

We transform the network into a tree-like structure, combining nodes into bags if needed, then perform message passing on this tree.

Junction tree algorithm Steffen Lauritzen (with David Spiegelhalter)

Can we do computations smarter?

- This computation can blow up quickly – can we do any faster? Yes, in several distinct ways!

We transform a Bayesian network into a polynomial function and perform inference by combining partial derivatives of this function.

Adnan Darwiche's differential algorithm

We look at the computations and find a smart way of reordering them, thus minimizing the number of summations and multiplications.

Rina Dechter's variable elimination algorithm

Variable elimination

- Variable elimination employs a different strategy than junction tree (think tree-decomposition)
- Try to solve a query by efficiently summing out of the variables (elimination) in a ‘logical’ order
- Works on arbitrary networks, not only trees
- We will need the concept of a **factor** to describe the algorithm – see last week, recap here

Useful notation: Factors

- A factor $f(X_1,\ldots,X_k)$:

$$f : X_1 \times \ldots \times X_k \rightarrow R$$

yields a real value ($r \in R$) for each concrete tuple

$$(x_1, \ldots, x_k) \in (X_1 \times \ldots \times X_k)$$

- Scope = $\{X_1,\ldots,X_k\}$ of “free variables”

From probability distributions to factors

- P(Tampering)
- P(Alarm)
- P(Report)
- P(Alarm | Tampering)
- P(Alarm | ¬tampering)
- P(¬alarm | Tampering)
- P(Smoke | Alarm)
- P(Report | Fire)
- …

Tampering	Prob
tampering	0.02
¬tampering	0.98

Alarm	Prob
alarm	0.0266
¬alarm	0.9734

Alarm	Tamp	Cond Prob
alarm	tamp	0.845
alarm	¬tamp	0.01
¬alarm	tamp	0.155
¬alarm	¬tamp	0.99

Factors

- $f_1(\text{Alarm, Tampering}) \overset{def}{=} P(\text{Alarm} \mid \text{Tampering})$
- $f_2(\text{Alarm}) \overset{def}{=} P(\text{Alarm} \mid \neg\text{tamp})$

Alarm	Tamp	f_1
alarm	tamp	0.845
alarm	¬tamp	0.01
¬alarm	tamp	0.155
¬alarm	¬tamp	0.99

Alarm	Tamp= ¬tamp	Cond Prob
~~alarm~~	~~tamp~~	~~0.845~~
alarm	¬tamp	0.01
~~¬alarm~~	~~tamp~~	~~0.155~~
¬alarm	¬tamp	0.99

Alarm	Tamp= ¬tamp	f_2
alarm	¬tamp	0.01
¬alarm	¬tamp	0.99

Factor product

- f_1(A,B) x f_2 (B,C) = f_3(A,B,C) where f_3(a,b,c) = f_1(a,b)*f_2(b,c) for all a ∈ A, b ∈ B and c ∈ C

f_1

a_1	b_1	0.5
a_1	b_2	0.8
a_2	b_1	0.1
a_2	b_2	0
a_3	b_1	0.3
a_3	b_2	0.9

x

f_2

b_1	c_1	0.5
b_1	c_2	0.7
b_2	c_1	0.1
b_2	c_2	0.2

=

f_3

a_1	b_1	c_1	0.5*0.5 = 0.25
a_1	b_1	c_2	0.5*0.7 = 0.35
a_1	b_2	c_1	0.8*0.1 = 0.08
a_1	b_2	c_2	0.8*0.2 = 0.16
a_2	b_1	c_1	0.1*0.5 = 0.05
a_2	b_1	c_2	0.1*0.7 = 0.07
a_2	b_2	c_1	0*0.1 = 0
a_2	b_2	c_2	0*0.2 = 0
a_3	b_1	c_1	0.3*0.5 = 0.15
a_3	b_1	c_2	0.3*0.7 = 0.21
a_3	b_2	c_1	0.9*0.1 = 0.09
a_3	b_2	c_2	0.9*0.2 = 0.18

Factor marginalization

- Summing out a factor: $$\sum_B f_3(A,B,C) = f_4(A,C)$$

f_3

a_1	b_1	c_1	0.5*0.5 = 0.25
a_1	b_1	c_2	0.5*0.7 = 0.35
a_1	b_2	c_1	0.8*0.1 = 0.08
a_1	b_2	c_2	0.8*0.2 = 0.16
a_2	b_1	c_1	0.1*0.5 = 0.05
a_2	b_1	c_2	0.1*0.7 = 0.07
a_2	b_2	c_1	0*0.1 = 0
a_2	b_2	c_2	0*0.2 = 0
a_3	b_1	c_1	0.3*0.5 = 0.15
a_3	b_1	c_2	0.3*0.7 = 0.21
a_3	b_2	c_1	0.9*0.1 = 0.09
a_3	b_2	c_2	0.9*0.2 = 0.18

f_4

a_1	c_1	0.25+0.08 = 0.33
a_1	c_2	0.35+0.16 = 0.51
a_2	c_1	0.05+0 = 0.05
a_2	c_2	0.07+0 = 0.07
a_3	c_1	0.15+0.09 = 0.24
a_3	c_2	0.21+0.18 = 0.39

Factor reduction

- $f_3(A,B,c_1) = f_5(A,B)$

f_3

a_1	b_1	c_1	0.5*0.5 = 0.25
a_1	b_1	c_2	0.5*0.7 = 0.35
a_1	b_2	c_1	0.8*0.1 = 0.08
a_1	b_2	c_2	0.8*0.2 = 0.16
a_2	b_1	c_1	0.1*0.5 = 0.05
a_2	b_1	c_2	0.1*0.7 = 0.07
a_2	b_2	c_1	0*0.1 = 0
a_2	b_2	c_2	0*0.2 = 0
a_3	b_1	c_1	0.3*0.5 = 0.15
a_3	b_1	c_2	0.3*0.7 = 0.21
a_3	b_2	c_1	0.9*0.1 = 0.09
a_3	b_2	c_2	0.9*0.2 = 0.18

f_5

a_1	b_1	0.25
a_1	b_2	0.08
a_2	b_1	0.05
a_2	b_2	0
a_3	b_1	0.15
a_3	b_2	0.09

Why factors?

- Fundamental building block for defining distributions in high-dimensional spaces
- Set of basic operations for manipulating these probability distributions
- We will use factors in our inference algorithm

Running example again

P(tampering) = 0.02

P(fire) = 0.01

P(alarm | fire $\wedge$ tampering) = 0.5

P(alarm | fire $\wedge$ $\neg$tampering) = 0.99

P(alarm | $\neg$fire $\wedge$ tampering) = 0.85

P(alarm | $\neg$fire $\wedge$ $\neg$tampering) = 0.0001

P(smoke | fire) = 0.9

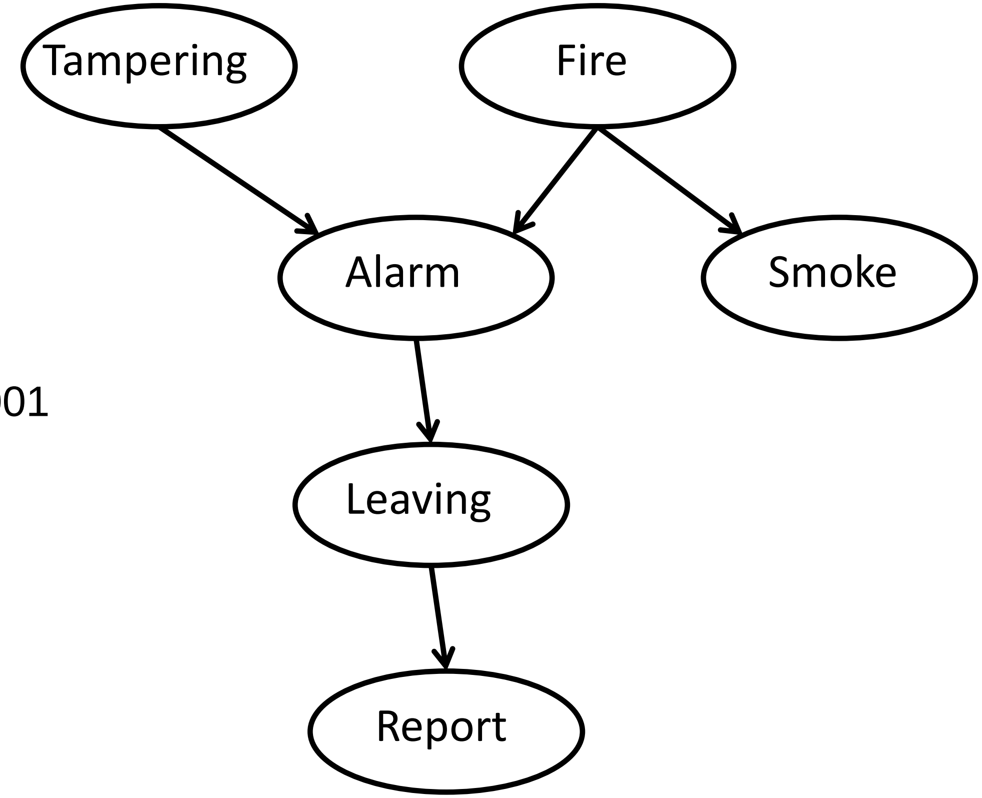
P(smoke | $\neg$fire) = 0.01

P(leaving | alarm) = 0.88

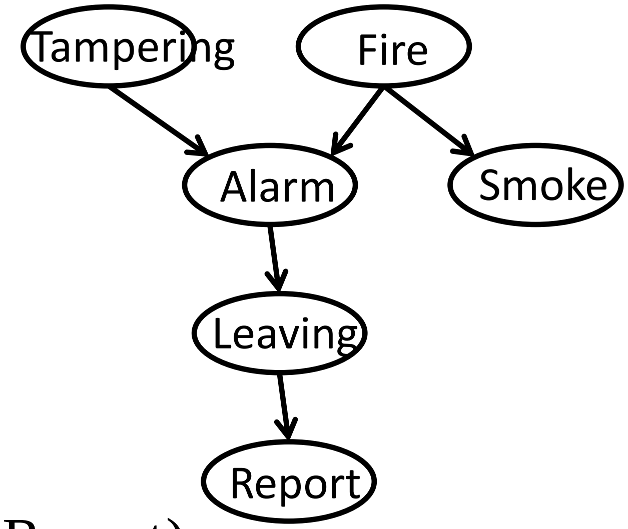
P(leaving | $\neg$alarm) = 0.001

P(report | leaving) = 0.75

P(report | $\neg$leaving) = 0.01



More complex computations



- Compute *P*(Report)

$$P(\text{Report}) = \sum_{\text{Tamp, Fire, Alarm, Smoke, Leaving}} P(\text{Tamper, Fire, Alarm, Smoke, Leaving, Report})$$

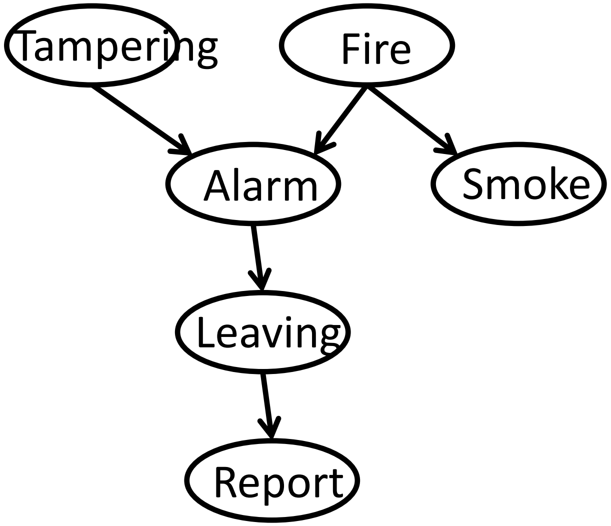
$$= \sum_{\text{Tamp, Fire, Alarm, Smoke, Leaving}} \{P(\text{Tamper})P(\text{Fire})P(\text{Alarm} \mid \text{Tamp, Fire})$$
$$P(\text{Smoke} \mid \text{Fire})P(\text{Leaving} \mid \text{Alarm})P(\text{Report} \mid \text{Leaving})\}$$

- Rearrange & sum the variables out in a clever order:

  Smoke – Fire – Tampering – Alarm – Leaving

More complex computations

- Compute P(Report)



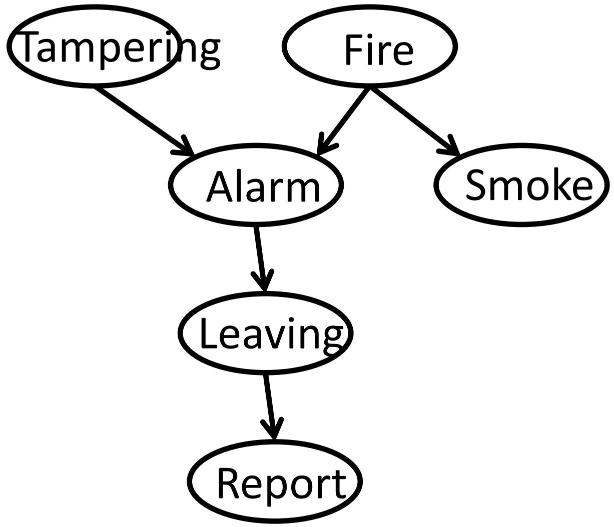
$$P(\text{Report}) =$$

$$\sum_{\text{Leaving}} \{P(\text{Report} \mid \text{Leaving}) \sum_{\text{Alarm}} \{P(\text{Leaving} \mid \text{Alarm}) \sum_{\text{Tamp}} \{P(\text{Tamp})$$

$$\sum_{Fire} \{P(\text{Fire}) P(\text{Alarm} \mid \text{Tamp}, \text{Fire}) \sum_{\text{Smoke}} P(\text{Smoke} \mid \text{Fire})\}\}\}\}$$

More complex computations

- Compute *P*(Report)



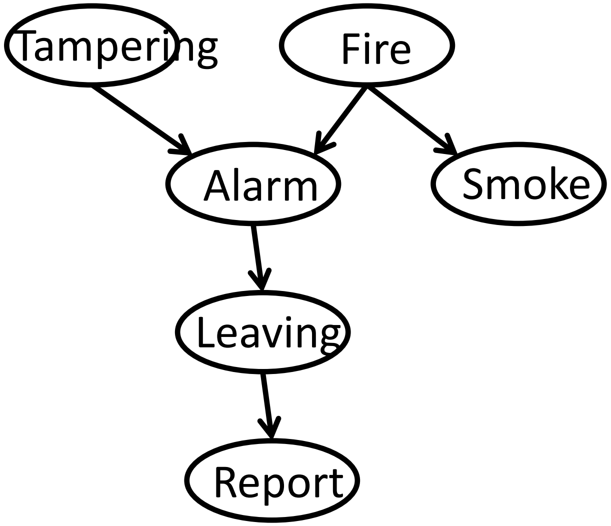
$$P(\text{Report}) =$$

$$\sum_{\text{Leaving}} \{ f(\text{Report}, \text{Leaving}) \sum_{\text{Alarm}} \{ f(\text{Leaving}, \text{Alarm}) \sum_{\text{Tamp}} \{ f(\text{Tamp})$$

$$\sum_{Fire} \{ f(\text{Fire}) f(\text{Alarm}, \text{Tamp}, \text{Fire}) \sum_{\text{Smoke}} f(\text{Smoke}, \text{Fire}) \} \} \} \}$$

$$f_1(\text{Fire}) = \sum_{\text{Smoke}} P(\text{Smoke} \mid \text{Fire}) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

More complex computations



- Compute *P*(Report)

$$P(\text{Report})$$

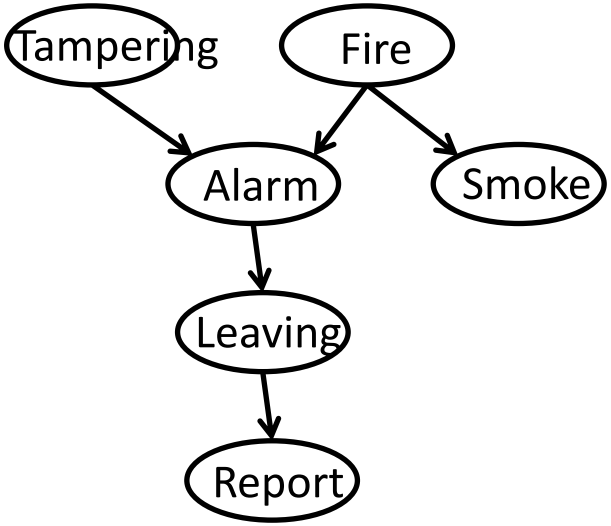
$$\sum_{\text{Leaving}} \{ f(\text{Report}, \text{Leaving}) \sum_{\text{Alarm}} \{ f(\text{Leaving}, \text{Alarm}) \sum_{\text{Tamp}} \{ f(\text{Tamp})$$

$$\sum_{\text{Fire}} \{ f(\text{Fire}) f(\text{Alarm}, \text{Tamp}, \text{Fire}) f_1(\text{Fire}) \} \} \} \}$$

$$\sum_{\text{Fire}} f(\text{Fire}) f(\text{Alarm}, \text{Tamp}, \text{Fire}) f_1(\text{Fire}) = f_2(\text{Alarm}, \text{Tamp})$$

More complex computations

- Compute *P*(Report)



$$P(\text{Report})$$

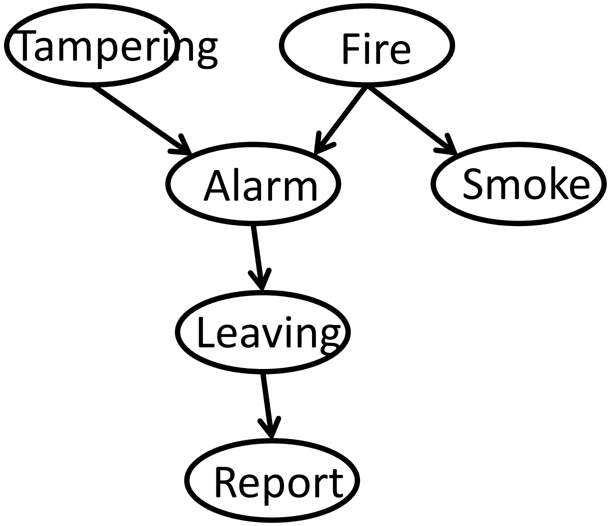
$$\sum_{\text{Leaving}} \{ f(\text{Report}, \text{Leaving}) \sum_{\text{Alarm}} \{ f(\text{Leaving}, \text{Alarm})$$

$$\sum_{\text{Tamp}} \{ f(\text{Tamp}) f_2(\text{Alarm}, \text{Tamp}) \} \} \}$$

$$\sum_{\text{Tamp}} f(\text{Tamp}) f_2(\text{Alarm}, \text{Tamp}) = f_3(\text{Alarm})$$

More complex computations

- Compute *P*(Report)



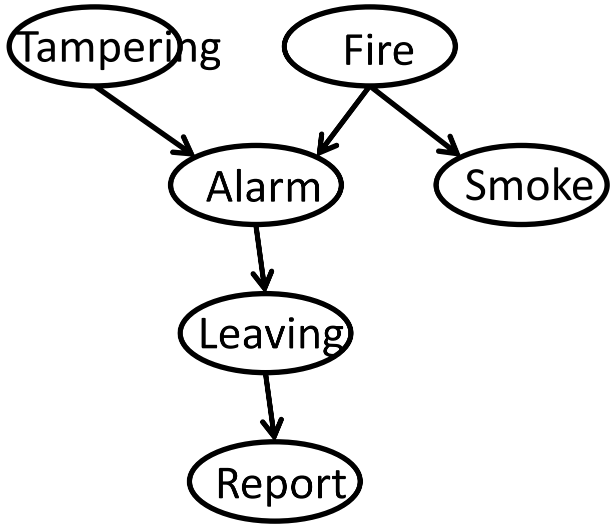
$$P(\text{Report})$$

$$\sum_{\text{Leaving}} \{ f(\text{Report}, \text{Leaving}) \sum_{\text{Alarm}} \{ f(\text{Leaving}, \text{Alarm}) f_3(\text{Alarm}) \} \}$$

$$\sum_{\text{Alarm}} \{ f(\text{Leaving}, \text{Alarm}) f_3(\text{Alarm}) \} = f_4(\text{Leaving})$$

More complex computations

- Compute P(Report)



$$P(\text{Report})$$

$$\sum_{\text{Leaving}} \{ f\,(\text{Report}, \text{Leaving}) f_4(\text{Leaving}) \} = f_5(\text{Report})$$

$$P(\text{Report}) = f_5(\text{Report}) / \sum_{\text{Report}} \{ f_5(\text{Report}) \}$$

Normalize!

Algorithm: Variable Elimination (VE)

- The algorithm we described is called **Variable Elimination (VE)** and can be implemented in a computer program (= Assignment 3)

- Based on the notion that a Bayesian network specifies a factorization of the joint probability distribution

- Computes with factors: functions of variables

Variable Elimination

a) What are the query variables?

b) What are the observed variables?

c) Write down:

    1) The product formula to compute the query

    2) The reduced formula based on the network structure

d) Identify factors and reduce observed variables

e) Fix an elimination ordering

f) For every variable Z in this ordering:

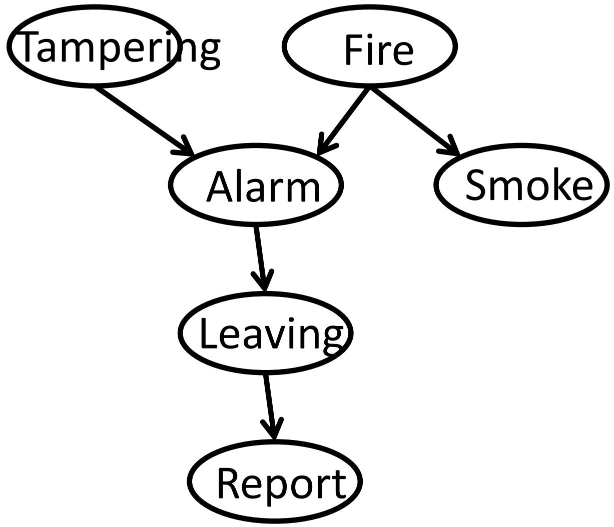
    a) Multiply factors containing Z

    b) Sum out Z to obtain new factor f_Z

    c) Remove the multiplied factors form the list and add f_Z

g) Normalize the result to make it a probability distribution

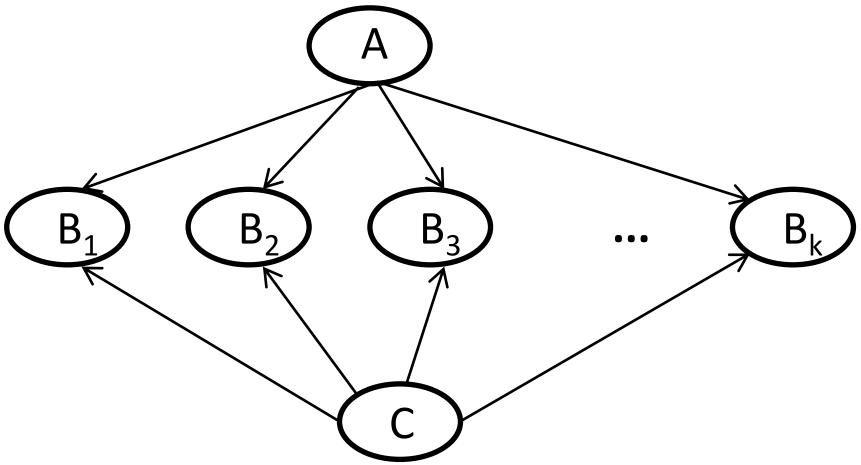
VE example



- Compute
  P(Tampering | smoke, report)
- What are the relevant factors?
- Which factors need to be eliminated?
  And in which order?
- How to renormalize in the end?
- Book section 8.4.1 – on the white board
- Knowledge clip on variable elimination

Complexity and elimination order

- Eliminate A first:
- $\sum_A g_1(A,B_1,\ldots, B_k, C)$
- Size of factor is **exponential in *k***

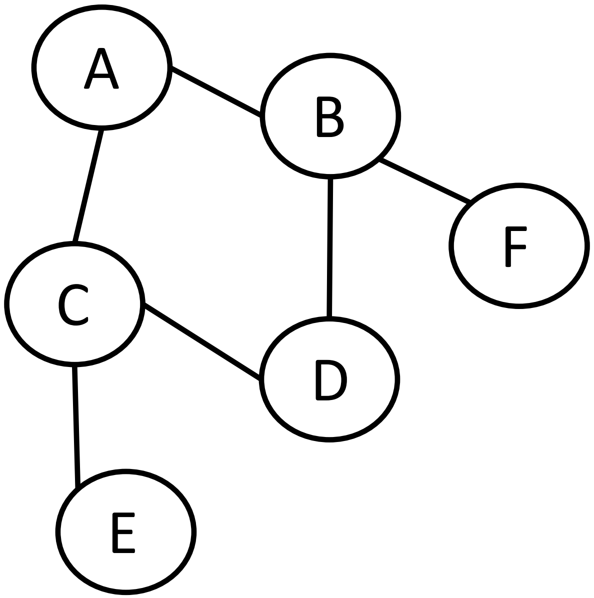


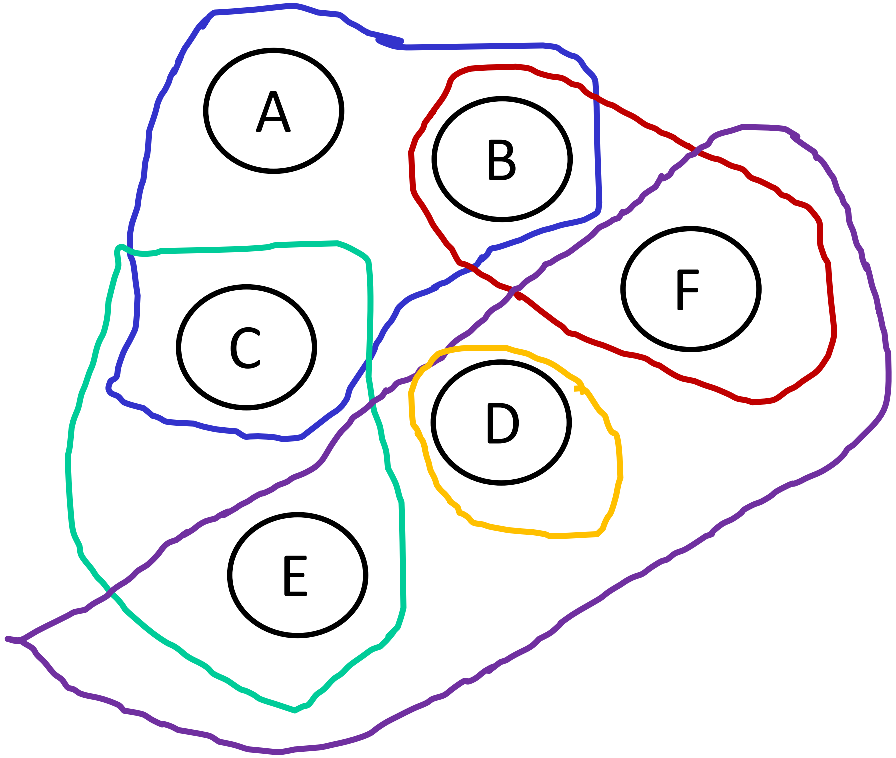
- Eliminate B's first:
- $\sum_{B1} g_1(A,B_1,C)$, $\sum_{B2} g_2(A,B_2,C)$, ...
- **linear in *k***

- Complexity of algorithm depends heavily on the choice of the **elimination ordering** – is worst case exponential

Hypergraph

- Extension of a ‘normal’ undirected graph where *hyper-edges* connect arbitrary number of vertices

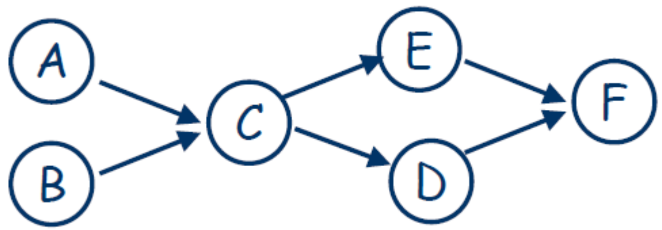


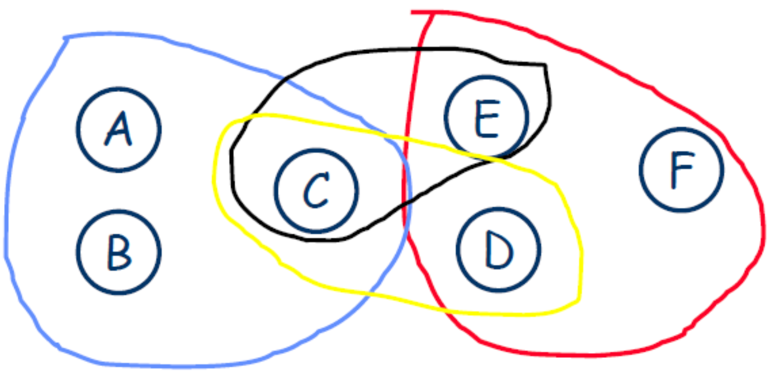


Hypergraph of a Bayesian network

- *Hypergraph of a Bayesian network:* one *hyper-edge* per CPT including every variable in the CPT

$$
\begin{aligned}
\Pr(A,B,C,D,E,F) = \; & \Pr(A)\Pr(B) \\
\text{X} \quad & \Pr(C|A,B) \\
\text{X} \quad & \Pr(E|C) \\
\text{X} \quad & \Pr(D|C) \\
\text{X} \quad & \Pr(F|E,D).
\end{aligned}
$$

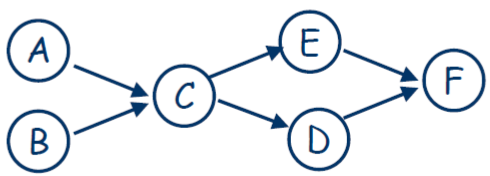




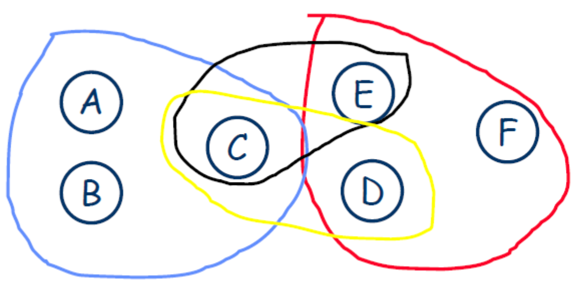
Variable elimination on hypergraph structure

- Elimination of variable X by:
  - Removing X from the graph
  - Creating a new hyper-edge = union of previous hyper-edges containing X
  - Removing original hyper-edges containing X

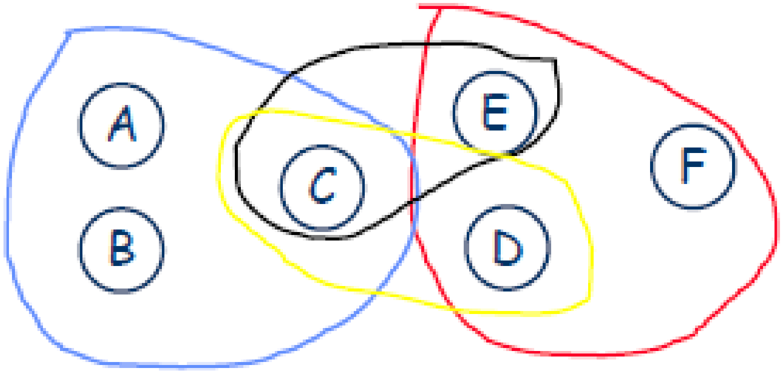
Pr(A,B,C,D,E,F) =
    Pr(A)Pr(B)
X   Pr(C|A,B)
X   Pr(E|C)
X   Pr(D|C)
X   Pr(F|E,D).



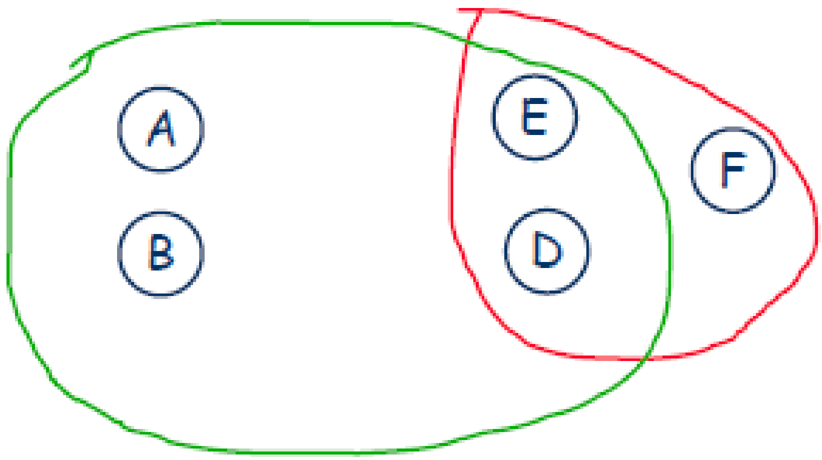
- Elimination of C?
- Elimination of D?
- Elimination of A?



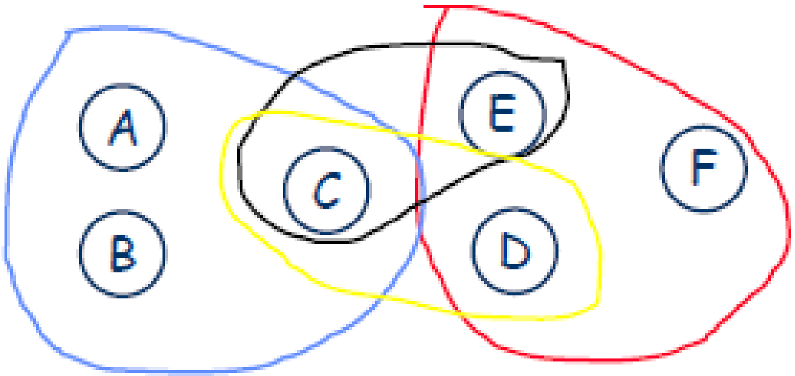
Variable elimination on hypergraph structure



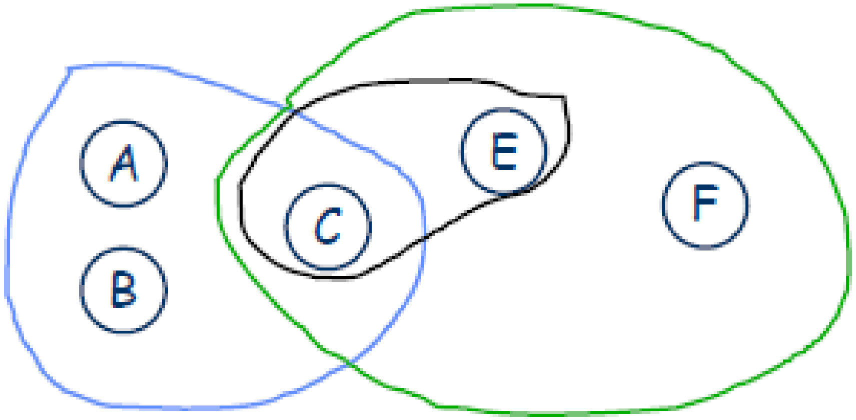
From above
Eliminate C



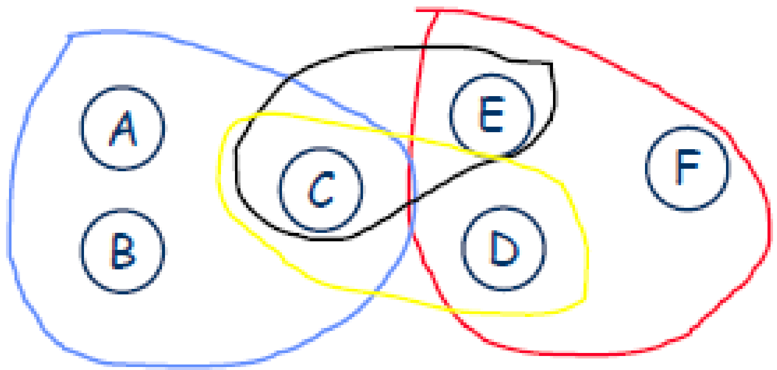
Variable elimination on hypergraph structure



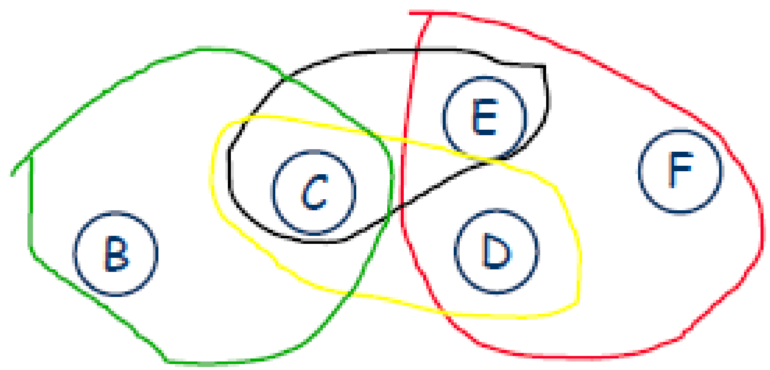
From above
Eliminate D



Variable elimination on hypergraph structure

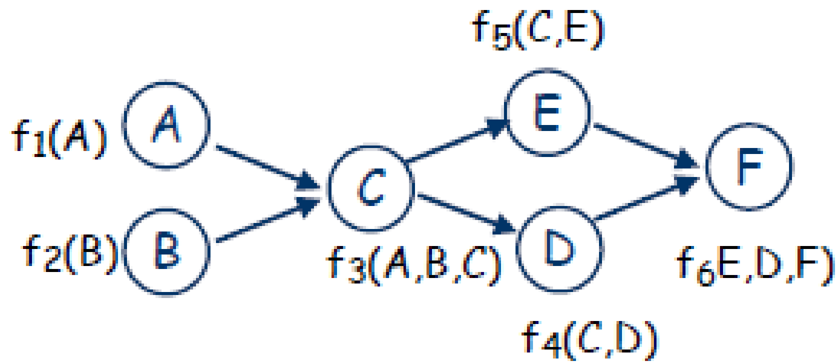


From above
Eliminate A

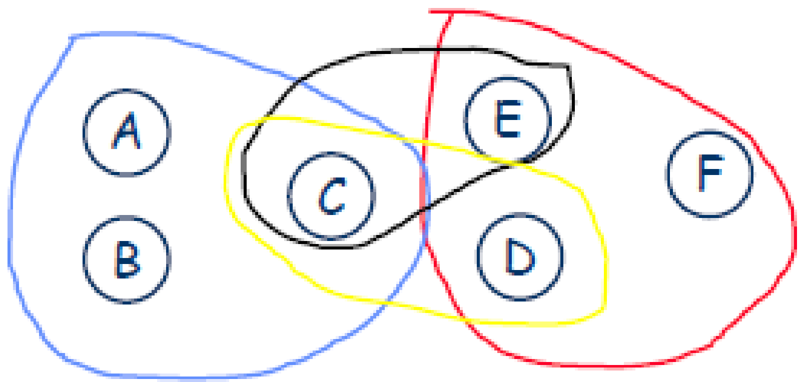


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

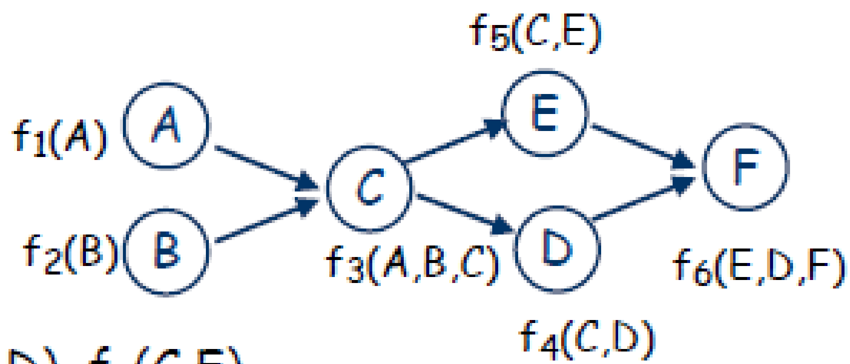


1. C:
2. F:
3. A:
4. B:
5. E:
6. D:

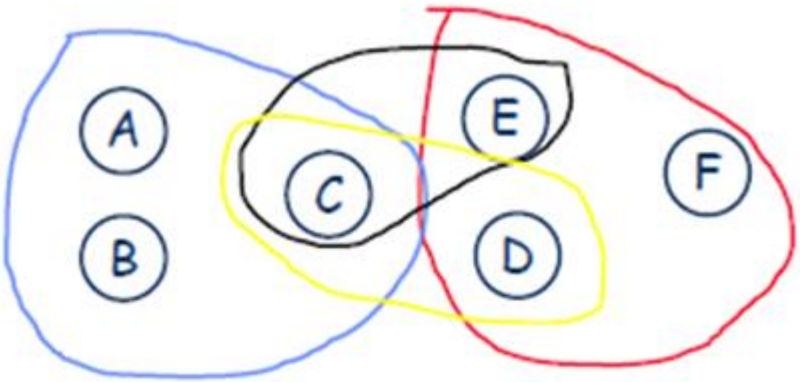


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

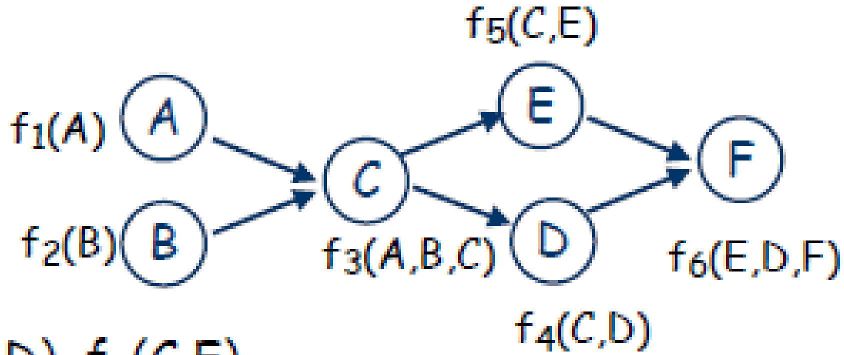


1. C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$
4. B: $f_2(B)$
5. E:
6. D:

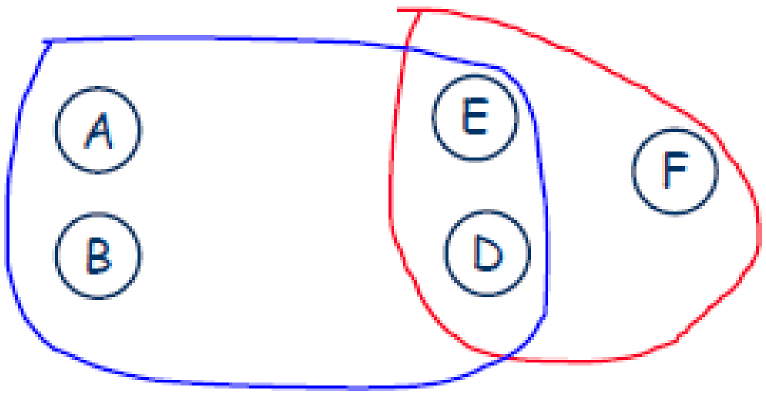


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

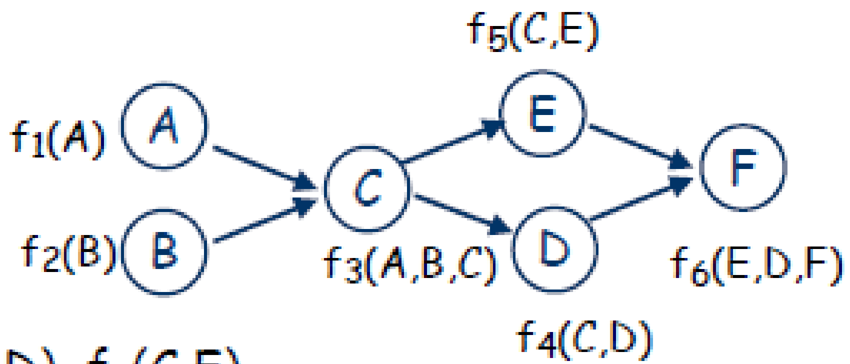


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. F: $f_6(E,D,F)$
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E:
6. D:

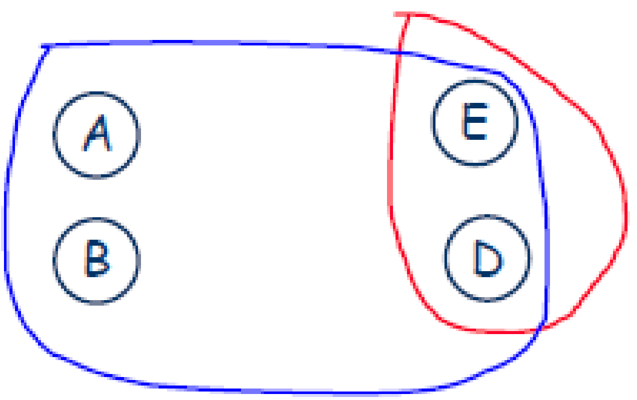


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

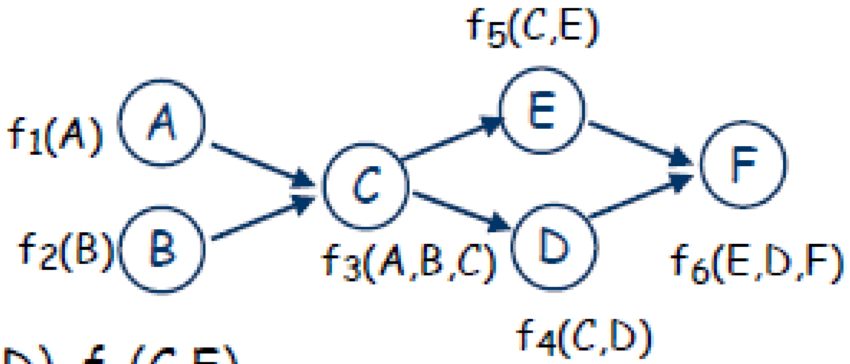


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. A: $f_1(A)$, $f_7(A,B,D,E)$
4. B: $f_2(B)$
5. E: $f_8(E,D)$
6. D:

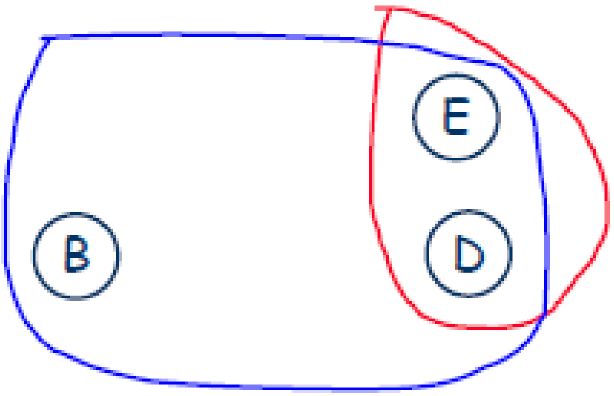


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

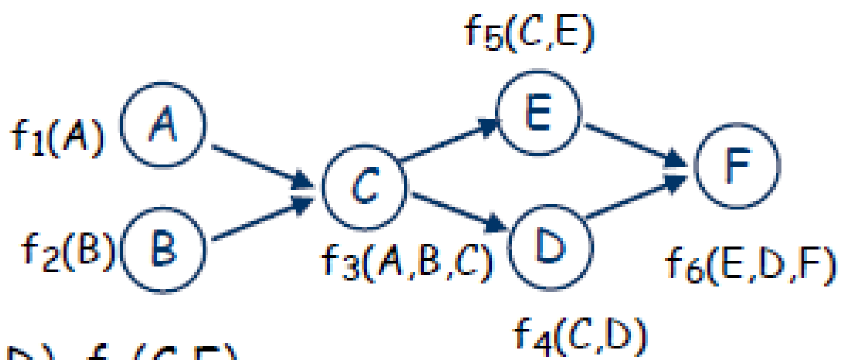


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. B: $f_2(B)$, $f_9(B,D,E)$
5. E: $f_8(E,D)$
6. D:

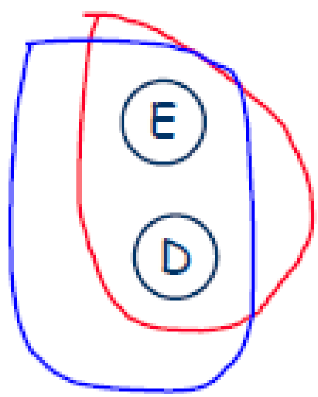


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D

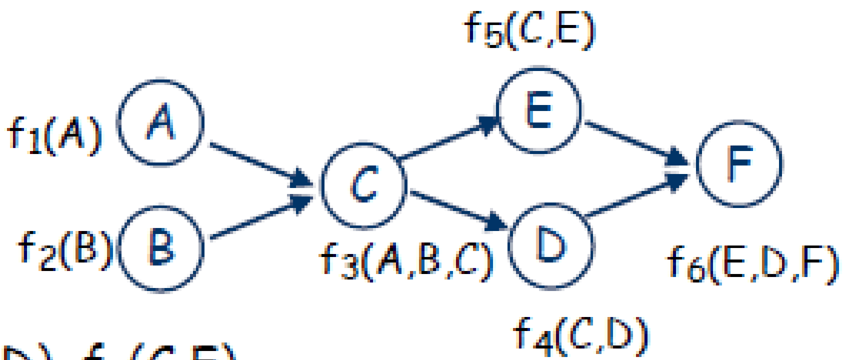


1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. E: $f_8(E,D)$, $f_{10}(D,E)$
6. D:

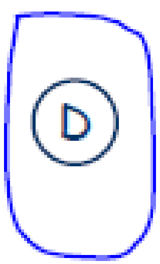


Variable elimination on hypergraph structure

Ordering:
C,F,A,B,E,D



1. ~~C: $f_3(A,B,C)$, $f_4(C,D)$, $f_5(C,E)$~~
2. ~~F: $f_6(E,D,F)$~~
3. ~~A: $f_1(A)$, $f_7(A,B,D,E)$~~
4. ~~B: $f_2(B)$, $f_9(B,D,E)$~~
5. ~~E: $f_8(E,D)$, $f_{10}(D,E)$~~
6. D: $f_{11}(D)$



Elimination width

- Given an ordering π of the variables and an initial hypergraph $\mathcal{H}$, eliminating these variables yields a sequence of hypergraphs

  $\mathcal{H}$ = H0, H1, H2, …, Hn,

  where Hn contains only one vertex (query variable)

- The elimination width of π is the maximum size (number of variables) of **any** hyper-edge in **any** of the hypergraphs H0, H1, H2, …, Hn

- The elimination width of the previous example was 4 ({A,B,E,D} in H1 and H2)

Elimination width

- If the elimination width of an ordering π is k, then the complexity of VE using that ordering is $2^{O(k)}$
- Elimination width k means that at some stage in the elimination process a factor involving k variables was generated
  - That factor will require $2^{O(k)}$ space to store
    - *Space complexity of VE is $2^{O(k)}$*
  - And it will require $2^{O(k)}$ operations to process
    - *Time complexity of VE is $2^{O(k)}$*
- Note that k is the elimination width of this **particular** ordering!

Elimination width and Tree-width

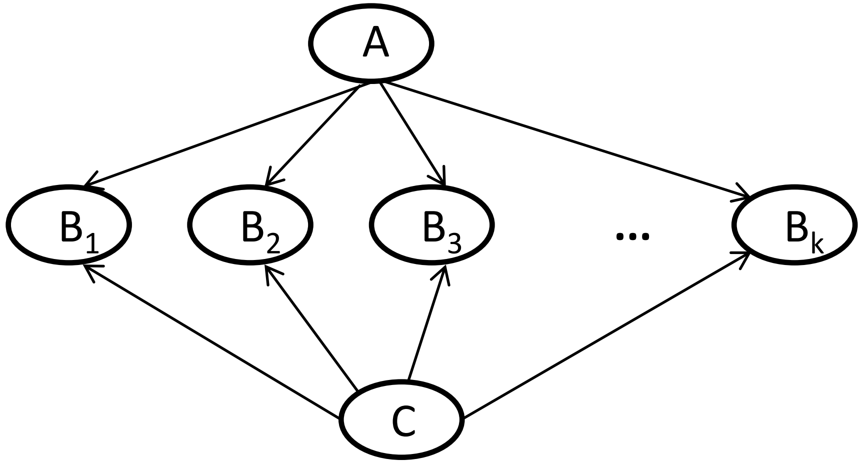
- The tree-width of a Bayesian network with *n* variables is the minimum elimination width of all its (*n* - 1)! elimination orderings, minus 1
- Variable elimination takes time / space, **exponential** in the tree-width of the Bayesian network, even for orderings with the minimal elimination width!
- In fact, this holds for all inference algorithms; limiting the tree-width of the network is both a *sufficient* and a *necessary* constraint for efficient computation

Johan Kwisthout, Hans Bodlaender, and Linda van der Gaag (2010). *The Necessity of Bounded Treewidth for Efficient Inference in Bayesian Networks*. Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10). IOS Press, pp. 237-242.

Finding good elimination orderings

- For a Bayesian network B, finding the **optimal** elimination ordering is NP-hard

- *Min-fill heuristic:* always eliminate next the variable that creates the smallest size factor.



Compute P(C):

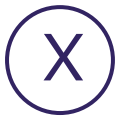
- A creates factor of size *k*
- Each B_i creates factor of size 2

Bayesian inference is NP-hard

- SAT: given a Boolean formula φ, is there a truth assignment to all variables that makes φ true?
- SAT Example: $\varphi = \neg(X_1 \vee X_2) \vee \neg X_3$
- Satisfiable with, e.g., $X_1 = T$, $X_2 = X_3 = F$
- We construct in poly time a Bayesian network B_φ from φ with a designated variable V_φ such that $P(V_\varphi = T) > 0$ if and only if φ is satisfiable
- Check that this proves NP-hardness!

Hardness proof constructs

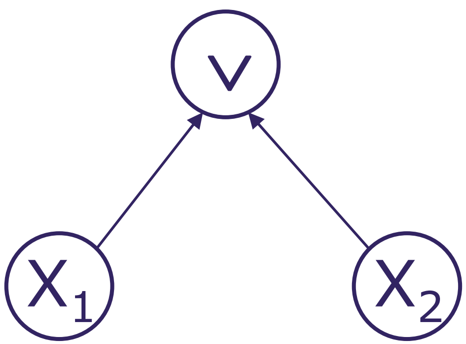
- Variables in φ are nodes in B_φ with values T, F (uniform probability)



$\Pr(X = T) = 0.5$

$\Pr(X = F) = 0.5$

- Operators in φ are nodes in B_φ with values T, F (probability table = truth value of logical component)



$\Pr(\vee = T \mid X_1 = T \text{ and } X_2 = T) = 1$

$\Pr(\vee = T \mid X_1 = T \text{ and } X_2 = F) = 1$

$\Pr(\vee = T \mid X_1 = F \text{ and } X_2 = T) = 1$

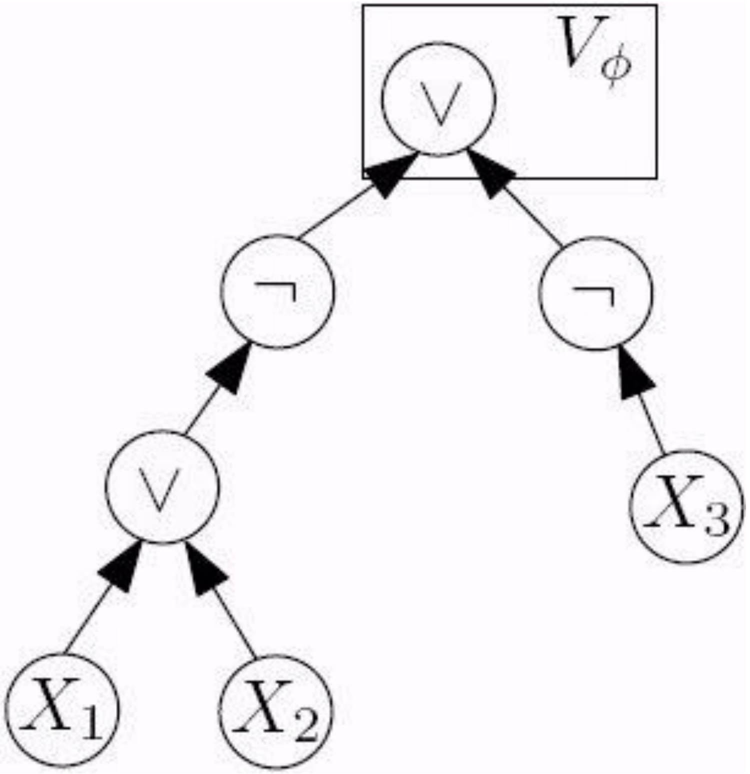
$\Pr(\vee = T \mid X_1 = F \text{ and } X_2 = F) = 0$

Constructing B_φ from a formula φ

- Start with adding the X-variables – one for each variable in φ
- Work ‘from the inside out’ (just as you would do when you evaluate the formula), i.e., start with the connectives that are the most deeply nested. Add variables corresponding to these connectives and connect the variables that are bound by them
- For the subsequent levels, you may need to connect ‘connective-variables’, rather than ‘variable-variables’. These ‘connective-variables’ represent sub-formula, rather than a single variable
- Negations take a single input, other connectives take two
- The top-level connective (with no outputs) is denoted by V_φ

Hardness proof constructs

$\varphi = \neg(X_1 \vee X_2) \vee \neg X_3$



$\Pr(V_\varphi = T \mid X_1 = T \wedge X_2 = T \wedge X_3 = T) = 0$

$\Pr(V_\varphi = T \mid X_1 = T \wedge X_2 = T \wedge X_3 = F) = 1$

:

$\Pr(V_\varphi = T \mid X_1 = F \wedge X_2 = F \wedge X_3 = F) = 1$

Now, $\Pr(V_\varphi = T) > 0$
if and only if there is a truth assignment that satisfies φ!

Bayesian inference is NP-hard

- Reduction takes polynomial time: we need one binary variable for each Boolean variable and one binary variable for each operator in the network

- This proves that Bayesian inference is NP-hard

- Question: is the following problem in NP?

  Given a Bayesian network B with variable V, what is P(V = T)?

Not a decision problem!

Bayesian inference is NP-hard

- Reduction takes polynomial time: we need one binary variable for each Boolean variable and one binary variable for each operator in the network

- This proves that Bayesian inference is NP-hard

- Question: is the following problem in NP?

  Given a Bayesian network B with variable V **and a rational number *q***, is P(V = T) > *q*?

Not verifiable in poly time!

Bayesian inference is NP-hard

- Reduction takes polynomial time: we need one binary variable for each Boolean variable and one binary variable for each operator in the network

- This proves that Bayesian inference is NP-hard

- Question: is the following problem in NP?

  Given a Bayesian network B with variable V,
  is P(V = T) > **0**?

  ✓ **One non-zero instantiation will do**

Important highlights in this lecture

- Naive inference computations
  - Compute simple inference queries using probability axioms
- Variable elimination algorithm
  - Understand, and be able to apply, the variable elimination algorithm for arbitrary networks and queries
  - Understand why the elimination order is important
  - Understand factors and operations on them
  - Being able to implement variable elimination
- Complexity
  - Understand why the Inference problem is NP-hard
  - Understand hyperedges and elimination width