CS486/686: Introduction to Artificial Intelligence
Lecture 7b - Bayesian Networks

Jesse Hoey & Victor Zhong

School of Computer Science, University of Waterloo

February 5, 2025

Readings: Poole & Mackworth Chap. 9-9.5

Review: Semantics of a Bayes' Net

The structure of the BN means that:

every X_i is **conditionally independent** of all its **nondescendants** given its parents:

$$P(X_i|S, Parents(X_i)) = P(X_i|Parents(X_i))$$

for any subset $S \subseteq NonDescendants(X_i)$

The BN defines a **factorization** of the **joint probability** distribution. The joint distribution is formed by multiplying the conditional probability tables together.

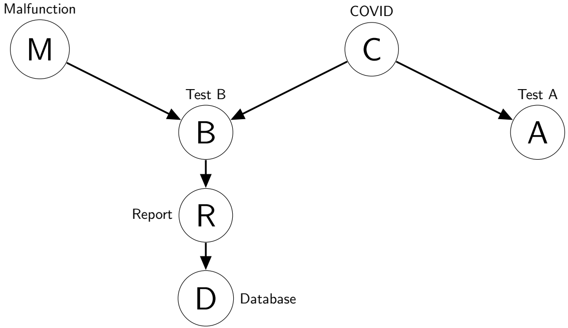
$$P(X_1, X_2, \ldots, X_n) = \prod_i P(X_i|parents(X_i))$$

Constructing Belief Networks

To represent a domain in a belief network, you need to consider:

- What are the **relevant variables**?
  - What will you observe? - This is the **evidence**
  - What would you like to find out? - This is the **query**
  - What other features make the model simpler? - These are the other variables
- What **values** should these variables take?
- What is the **relationship** between them? This should be expressed in terms of **local influence**.
- How does the value of each variable **depend on its parents**? This is expressed in terms of the **conditional probabilities**.

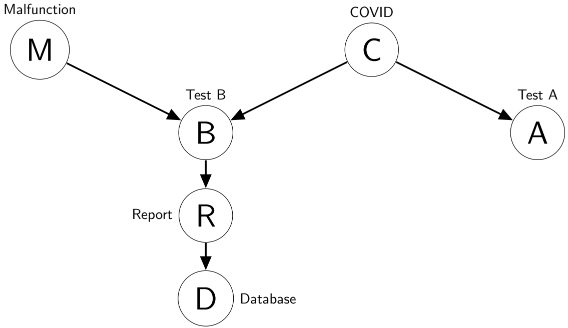
Bayesian Networks - Independence Assumptions



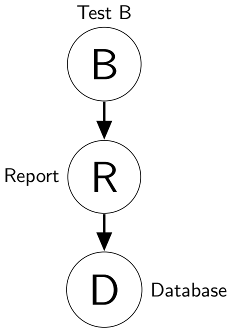
- **Test B** depends on **COVID** and **Malfunction**
- **Test A** depends only on **COVID**
- **Report** depends only on **Test B**
- **Database** depends only on **Report**

What are the **independencies?**

Three Basic Bayesian Networks

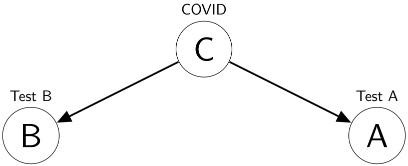


Three Basic Bayesian Networks



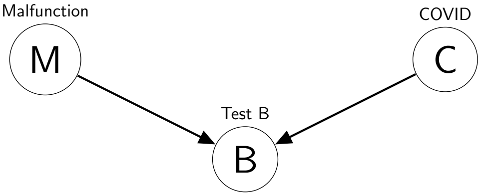
Database and Test B independent if Report is observed

Three Basic Bayesian Networks



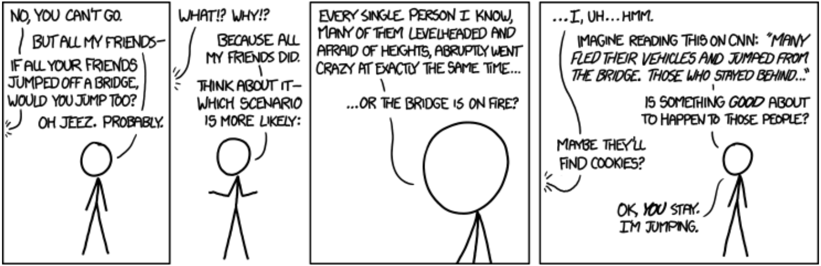
Test B and Test A are independent if COVID is observed

Three Basic Bayesian Networks

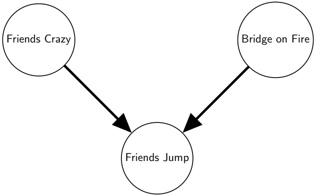


Malfunction and COVID are independent if Test B is not observed

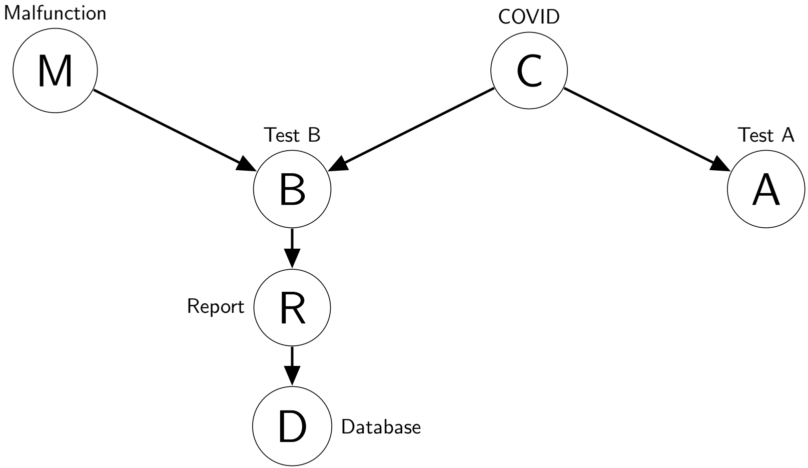
Three Basic Bayesian Networks



http://imgs.xkcd.com/comics/bridge.png



Three Basic Bayesian Networks...Recap



Updating belief: Bayes' Rule

Agent has a **prior belief** in a **hypothesis**, h, $P(h)$,

Agent observes some **evidence** e

that has a **likelihood** given the hypothesis: $P(e|h)$.

The agent's **posterior belief** about h after observing e, $P(h|e)$,

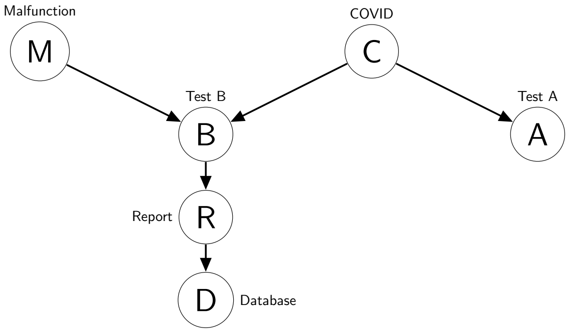
is given by **Bayes' Rule:**

$$P(h|e) = \frac{P(e|h)P(h)}{P(e)} = \frac{P(e|h)P(h)}{\sum_h P(e|h)P(h)}$$

Why is Bayes' theorem interesting?

- Often you have **causal knowledge**:
  $P(\mathit{symptom} \mid \mathit{disease})$
  $P(\underline{\text{light is off}} \mid \underline{\text{status of switches and switch positions}})$
  $P(\mathit{alarm} \mid \mathit{fire})$
  $P($image looks like  $\mid$ a tree is in front of a car$)$
- And want to do **evidential reasoning**:
  $P(\mathit{disease} \mid \mathit{symptom})$
  $P(\underline{\text{status of switches}} \mid \underline{\text{light is off and switch positions}})$
  $P(\mathit{fire} \mid \mathit{alarm})$
  $P($a tree is in front of a car $\mid$ image looks like $)$

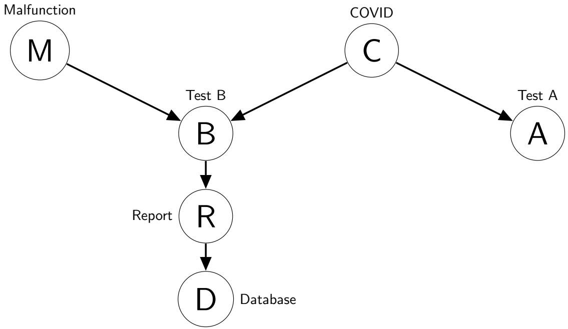
Probabilistic Inference



Before you get any information

- $P(COVID) = 0.32$
- $P(Malfunction) = 0.08$

Probabilistic Inference

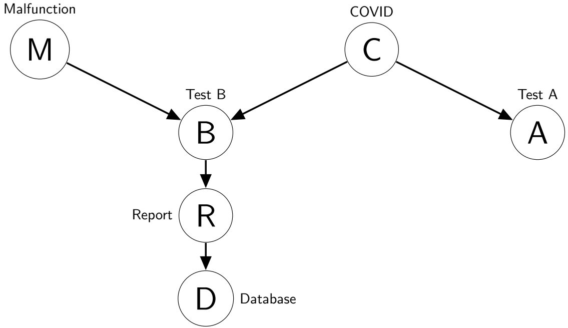


Suppose the doctor reads a **positive Test B in the Database**
evidence gives Database=true (not directly Test B= true)
we want to know $P(COVID = true|Database = true)$

- $P(COVID = true|Database = true) = 0.80$
- $P(Malfunction = true|Database = true) = 0.14$

(we will see how to get these numbers later)

Probabilistic Inference



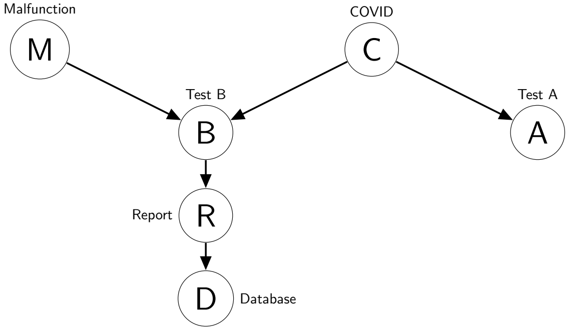
Suppose **Test A is positive** as well

we want $P(COVID = true | Database = true \wedge TestA = true)$

- $P(COVID = true | Database = true \wedge TestA = true) = 0.95$
- $P(M = true | Database = true \wedge TestA = true) = 0.08$

(we will see how to get these numbers later)

Probabilistic Inference



Suppose **Test A is negative**, though!

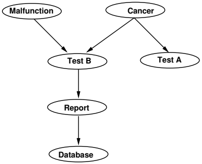
we want $P(COVID = true | Database = true \wedge TestA = false)$

- $P(COVID = true | Database = true \wedge TestA = false) = 0.48$
- $P(M = true | Database = true \wedge TestA = false) = 0.27$

(we will see how to get these numbers later)

Simple Forward Inference (Chain)

Computing marginal requires simple forward propagation of probabilities

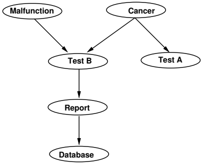


- $P(B) = \sum_{m,c} P(M = m, C = c, B)$ (marginalization - sum rule)
- $P(B) = \sum_{m,c} P(B \mid m, c)P(m \mid c)P(c)$ (chain rule)
- $P(B) = \sum_{m,c} P(B \mid m, c)P(m)P(c)$ (independence)
- $P(B) = \sum_{m} P(m) \sum_{c} P(c)P(B \mid m, c)$ (distribution of product over sum)

Note: all terms on the last line are CPTs in the BN
Note: only ancestors of B are considered. Why?

Simple Forward Inference (Chain)

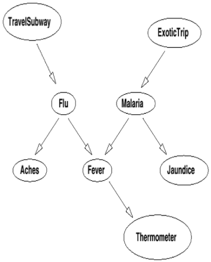
Same idea when evidence $COVID = true$ (denoted by c) "upstream"



- $P(R \mid c) = \sum_{m,b} P(R, b, m \mid c)$ (marginalization)
- $P(R \mid c) = \sum_{m,b} P(R \mid b, m, c) P(b \mid m, c) P(m \mid c)$ (chain rule)
- $P(R \mid c) = \sum_{m,b} P(R \mid b) P(b \mid m, c) P(m)$ (independence and conditional independence)

Simple Forward Inference

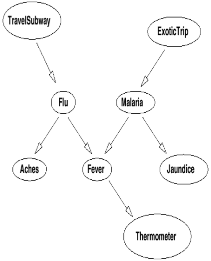
With multiple parents the evidence is "pooled"



$$
\begin{aligned}
P(Fev) &= \sum_{Flu,M,TS,ET} P(Fev, Flu, M, TS, ET) \\
&= \sum_{Flu,M} P(Fev|M, Flu) \\
& \quad \left[\sum_{TS} P(Flu|TS)P(TS)\right] \\
& \quad \left[\sum_{ET} P(M|ET)P(ET)\right]
\end{aligned}
$$

Simple Forward Inference

Also works with "upstream" evidence



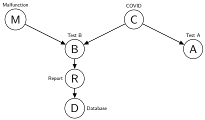
$$P(Fev|ts,\overline{m}) = \sum_{Flu} P(Fev, Flu|\overline{m}, ts)$$

$$= \sum_{Flu} P(Fev|Flu, ts, \overline{m})P(Flu|ts, \overline{m})$$

$$= \sum_{Flu} P(Fev|Flu, \overline{m})P(Flu|ts)$$

Simple Backward Inference

When evidence is downstream of query, then we must reason "backwards," which requires Bayes' rule



$$P(B \mid r) = P(r \mid B)P(B)/P(r) \propto P(r, B)$$

$$\begin{aligned}
P(r, B) &= \sum_{m,c} P(m, c, B, r) \\
&= \sum_{m,c} P(m)P(c \mid m)P(B \mid m, c)P(r \mid B, m, c)
\end{aligned}$$

(marginalization)

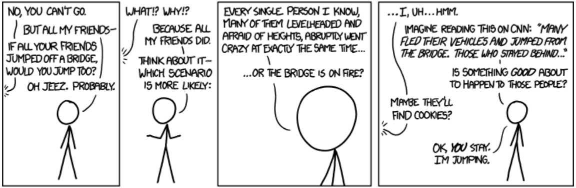
$$= \sum_{m,c} P(m)P(c)P(B \mid m, c)P(r \mid B)$$

(independence and conditional independence)

Normalizing constant is $\frac{1}{P(r)}$, but this can be computed as

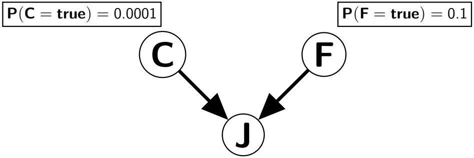
$$P(r) = \sum_{b} P(r, b)$$

Backward Inference



`http://imgs.xkcd.com/comics/bridge.png`

F: Bridge on Fire
C: All friends Crazy
J: All friends Jump
What is $P(F|J = true)$?



$\mathbf{P}(\mathbf{J} = \mathbf{true}|\mathbf{F}, \mathbf{C}) =$

F	C	$P(J = true\|F, C)$
t	t	0.95
t	f	0.99
f	t	0.99
f	f	0.01

Variable Elimination

- Intuitions above: **polytree** algorithm
- Works for simple networks without loops
- More general algorithm: **Variable Elimination**
- Applies sum-out rule repeatedly
- Distributes sums

Factors

A **factor** is a representation of a function from a tuple of random variables into a number.

We will write factor f on variables $X_1, \ldots, X_j$ as $f(X_1, \ldots, X_j)$.

We can assign some or all of the variables of a factor

$\rightarrow$(this is **restricting** a factor):

- $f(X_1 = v_1, X_2, \ldots, X_j)$, where $v_1 \in dom(X_1)$, is a **factor on** $X_2, \ldots, X_j$.
- $f(X_1 = v_1, X_2 = v_2, \ldots, X_j = v_j)$ is a number that is the **value of** f when each X_i has value v_i.

The former is also written as $f(X_1, X_2, \ldots, X_j)_{X_1 = v_1}$, etc.

Example Factors - Restricting a Factor

$r(X, Y, Z)$:

X	Y	Z	val
t	t	t	0.1
t	t	f	0.9
t	f	t	0.2
t	f	f	0.8
f	t	t	0.4
f	t	f	0.6
f	f	t	0.3
f	f	f	0.7

$r(X{=}t, Y, Z)$:

Y	Z	val
t	t	0.1
t	f	0.9
f	t	0.2
f	f	0.8

$r(X{=}t, Y, Z{=}f)$:

Y	val
t	0.9
f	0.8

$$r(X{=}t, Y{=}f, Z{=}f) = 0.8$$

Multiplying Factors

The **product** of factor $f_1(X, Y)$ and $f_2(Y, Z)$, where Y are the variables in common, is the factor $(f_1 \times f_2)(X, Y, Z)$ defined by:

$$(f_1 \times f_2)(X, Y, Z) = f_1(X, Y) f_2(Y, Z).$$

Multiplying Factors: Example

f_1:

A	B	val
t	t	0.1
t	f	0.9
f	t	0.2
f	f	0.8

f_2:

B	C	val
t	t	0.3
t	f	0.7
f	t	0.6
f	f	0.4

$f_1 \times f_2$:

A	B	C	val
t	t	t	0.03
t	t	f	0.07
t	f	t	0.54
t	f	f	0.36
f	t	t	0.06
f	t	f	0.14
f	f	t	0.48
f	f	f	0.32

Summing out Variables

We can **sum out** a variable, say X_1 with domain $\{v_1, \ldots, v_k\}$, from factor $f(X_1, \ldots, X_j)$, resulting in a factor on $X_2, \ldots, X_j$ defined by:

$$\begin{aligned}
&(\sum_{X_1} f)(X_2, \ldots, X_j) \\
&\quad = \quad f(X_1 = v_1, \ldots, X_j) + \cdots + f(X_1 = v_k, \ldots, X_j)
\end{aligned}$$

Summing out a Variable: Example

f_3:

A	B	C	val
t	t	t	0.03
t	t	f	0.07
t	f	t	0.54
t	f	f	0.36
f	t	t	0.06
f	t	f	0.14
f	f	t	0.48
f	f	f	0.32

$\sum_B f_3$:

A	C	val
t	t	0.57
t	f	0.43
f	t	0.54
f	f	0.46

Evidence

If we want to compute the posterior probability of Z given evidence $Y_1 = v_1 \wedge \ldots \wedge Y_j = v_j$:

$$
\begin{aligned}
& P(Z \mid Y_1 = v_1, \ldots, Y_j = v_j) \\
= \quad & \frac{P(Z, Y_1 = v_1, \ldots, Y_j = v_j)}{P(Y_1 = v_1, \ldots, Y_j = v_j)} \\
= \quad & \frac{P(Z, Y_1 = v_1, \ldots, Y_j = v_j)}{\sum_Z P(Z, Y_1 = v_1, \ldots, Y_j = v_j)}.
\end{aligned}
$$

The computation reduces to the **joint** probability of $P(Z, Y_1 = v_1, \ldots, Y_j = v_j)$.

normalize at the end.

can also restrict the query variable, e.g. compute: $P(Z = z \mid Y_1 = v_1, \ldots, Y_j = v_j)$

Probability of a Conjunction

Suppose the variables of the belief network are $X_1, \ldots, X_n$.

To compute $P(Z, Y_1 = v_1, \ldots, Y_j = v_j)$, we **sum out the variables other than query Z and evidence Y**,

$Z_1, \ldots, Z_k = \{X_1, \ldots, X_n\} - \{Z\} - \{Y_1, \ldots, Y_j\}$.

We order the Z_i into an **elimination ordering $Z_1 \ldots Z_k$.**

$$
\begin{aligned}
& P(Z, Y_1 = v_1, \ldots, Y_j = v_j) \\
= \quad & \sum_{Z_k} \cdots \sum_{Z_1} P(X_1, \ldots, X_n)_{Y_1 = v_1, \ldots, Y_j = v_j}. \\
= \quad & \sum_{Z_k} \cdots \sum_{Z_1} \prod_{i=1}^{n} P(X_i | parents(X_i))_{Y_1 = v_1, \ldots, Y_j = v_j}.
\end{aligned}
$$

Computing sums of products

Computation in belief networks reduces to **computing the sums of products**

- How can we compute $ab + ac$ efficiently?

Computing sums of products

Computation in belief networks reduces to **computing the sums of products**

- How can we compute $ab + ac$ efficiently?
- **Distribute** out the a giving $a(b + c)$

Computing sums of products

Computation in belief networks reduces to **computing the sums of products**

- How can we compute $ab + ac$ efficiently?
- **Distribute** out the a giving $a(b + c)$
- How can we compute $\sum_{Z_1} \prod_{i=1}^{n} P(X_i | parents(X_i))$ efficiently?

Computing sums of products

Computation in belief networks reduces to **computing the sums of products**

- How can we compute $ab + ac$ efficiently?
- **Distribute** out the a giving $a(b + c)$
- How can we compute $\sum_{Z_1} \prod_{i=1}^{n} P(X_i | parents(X_i))$ efficiently?
- Distribute out those factors that don't involve Z_1

Variable elimination algorithm

To compute $P(Z|Y_1 = v_1 \wedge \ldots \wedge Y_j = v_j)$:

- Construct a **factor for each conditional probability**.
- **Restrict** the observed variables to their observed values
- **Sum out** each of the other variables (the $\{Z_1, \ldots, Z_k\}$ from frame 23) according to some **elimination ordering**:
  for each Z_i in order starting from $i = 1$:
  - collect all factors that contain Z_i
  - multiply together and sum out Z_i
  - add resulting new factor back to the pool
- **Multiply** the remaining factors
- **Normalize** by dividing
  the resulting factor $f(Z)$ by $\sum_Z f(Z)$

Summing out a variable

To sum out a variable Z_j from a product $f_1, \ldots, f_k$ of factors:

- **Partition** the factors into
  - those that don't contain Z_j, say $f_1, \ldots, f_i$,
  - those that contain Z_j, say $f_{i+1}, \ldots, f_k$

We know:

$$\sum_{Z_j} f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \times \left(\sum_{Z_j} f_{i+1} \times \cdots \times f_k \right).$$

- **Explicitly construct** a representation of the rightmost factor $\left(\sum_{Z_j} f_{i+1} \times \cdots \times f_k \right)$.
- **Replace** the factors $f_{i+1}, \ldots, f_k$ by the new factor.

Example I

$\mathbf{P}(\mathbf{A} = \mathbf{true}) = 0.3$

$\mathbf{P}(\mathbf{C} = \mathbf{true} | \mathbf{A}) =$

A	$P(C = true \mid A)$
t	0.8
f	0.15

$\mathbf{P}(\mathbf{G} = \mathbf{true} | \mathbf{C}) =$

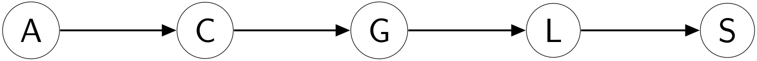
C	$P(G = true \mid C)$
t	1.0
f	0.2

$\mathbf{P}(\mathbf{L} = \mathbf{true} | \mathbf{G}) =$

G	$P(L = true \mid G)$
t	0.7
f	0.2

$\mathbf{P}(\mathbf{S} = \mathbf{true} | \mathbf{L}) =$

L	$P(S = true \mid L)$
t	0.9
f	0.3

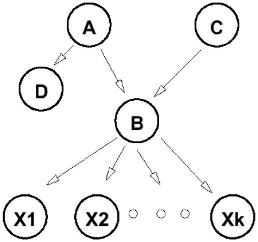


see note `variableelim.pdf`

Notes on VE

- Complexity is **linear** in number of variables, and **exponential** in the size of the largest factor
- When we create new factors: sometimes this **blows up**
- Depends on the **elimination ordering**
- For **polytrees**: work outside in
- For general BNs this can be hard
- simply **finding** the optimal elimination ordering is NP-hard for general BNs
- inference in general is NP-hard

Variable Ordering: Polytrees



- eliminate **singly-connected** nodes ($D, A, C, X_1, \ldots, X_k$) first
- Then no factor is ever larger than original CPTs
- If you eliminate B first, a **large factor** is created that includes $A, C, X_1, \ldots, X_k$

Variable Ordering: Relevance

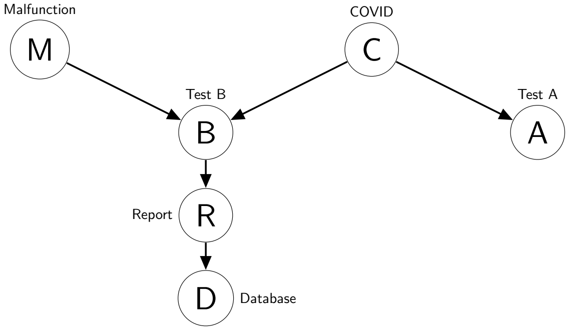
- Certain variables have **no impact**
- In ABC network above, computing $P(A)$ does **not require** summing over B and C

$$
\begin{aligned}
P(A) &= \sum_{B,C} P(C|B)P(B|A)P(A) \\
&= P(A)\sum_{B} P(B|A)\sum_{C} P(C|B) = P(A) * 1.0 * 1.0
\end{aligned}
$$

Variable Ordering: Relevance

- Can restrict attention to **relevant** variables:
- Given query Q and evidence $\mathbf{E}$, **complete** approximation is:
  - Q is relevant
  - if any node is relevant, its parents are relevant
  - if $E \in \mathbf{E}$ is a descendent of a relevant variable, then E is relevant
- irrelevant variable: a node that is not an ancestor of a query or evidence variable
- this will only remove irrelevant variables, but may not remove them all

Example II



see note `variableelim.pdf`

Next

Uncertainty (cont.): Advanced techniques in Modeling Uncertainty