# Lecture 21: 9 April, 2026

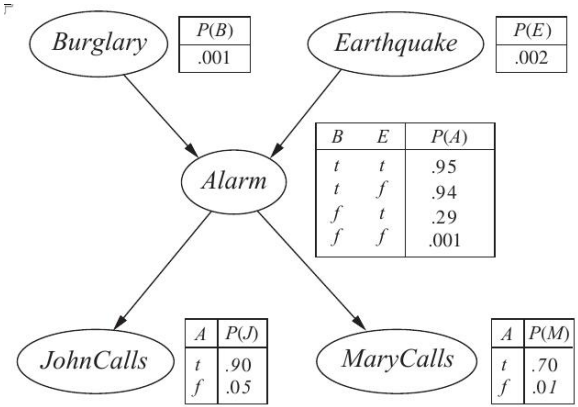
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–April 2026

# Probabilistic graphical models

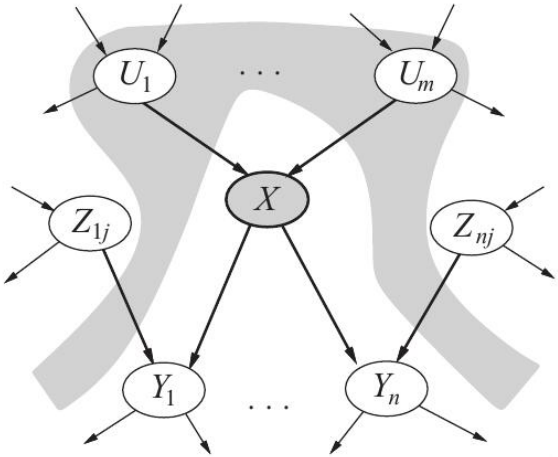
- Underlying DAG, no cyclic dependencies
- Each node has a local (conditional) probability table



# Probabilistic graphical models

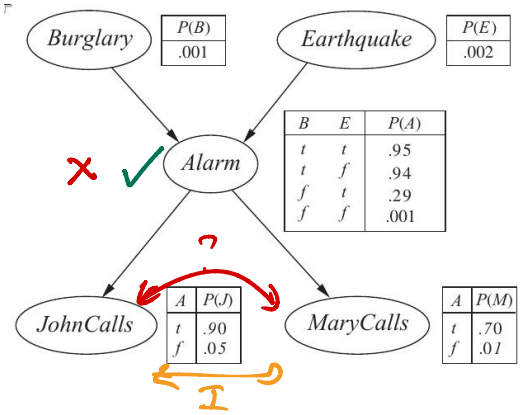
- Fundamental assumption

  A node is conditionally independent of non-descendants, given its parents



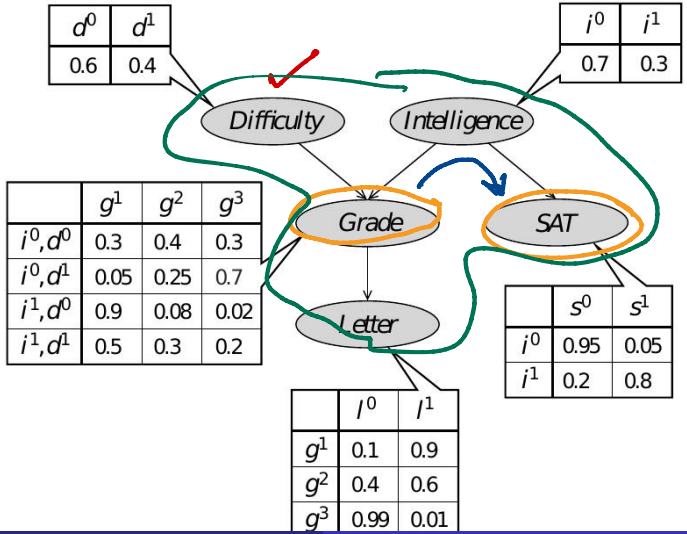
# Conditional independence

- $x \perp y$ — $x$ and $y$ are independent
  - $P(x \wedge y) = P(x) \cdot P(y)$
- $x \perp y \mid z$
  - $x$ and $y$ are independent given $z$
  - $P(x \wedge y \mid z) = P(x \mid z) \cdot P(y \mid z)$
- Is *JohnCalls* independent of *MaryCalls* ($j \perp m$)?
  - No — value of $j$ tells us something about value of $m$ and vice versa
- Is *JohnCalls* independent of *MaryCalls* given *Alarm* ($j \perp m \mid a$)?
  - Yes — by semantics of network, local independence



# Student example

- $SAT \perp Grade \mid Difficulty$ ?
  - No
- Can we calculate conditional independence from the graph?
- In general, check if $X \perp Y \mid Z$ for sets of variables $X$, $Y$, $Z$




| $d^0$ | $d^1$ |
|---|---|
| 0.6 | 0.4 |

| $i^0$ | $i^1$ |
|---|---|
| 0.7 | 0.3 |

| | $g^1$ | $g^2$ | $g^3$ |
|---|---|---|---|
| $i^0, d^0$ | 0.3 | 0.4 | 0.3 |
| $i^0, d^1$ | 0.05 | 0.25 | 0.7 |
| $i^1, d^0$ | 0.9 | 0.08 | 0.02 |
| $i^1, d^1$ | 0.5 | 0.3 | 0.2 |

| | $s^0$ | $s^1$ |
|---|---|---|
| $i^0$ | 0.95 | 0.05 |
| $i^1$ | 0.2 | 0.8 |

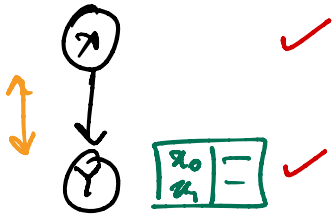
| | $l^0$ | $l^1$ |
|---|---|---|
| $g^1$ | 0.1 | 0.9 |
| $g^2$ | 0.4 | 0.6 |
| $g^3$ | 0.99 | 0.01 |

# Conditional independence

- How does dependence “flow” through a network?

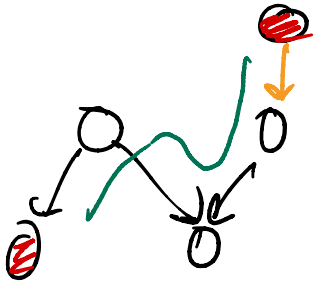
# Conditional independence

- How does dependence "flow" through a network?
- For neighbouring nodes, dependence flows both ways
  - If $x \to y$, knowing $x$ tells us about $y$ and vice versa



# Conditional independence

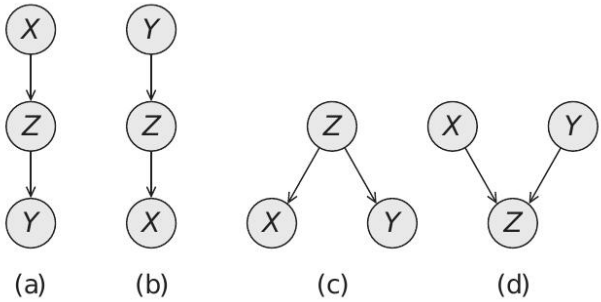
- How does dependence "flow" through a network?
- For neighbouring nodes, dependence flows both ways
  - If $x \to y$, knowing $x$ tells us about $y$ and vice versa
- Examine trails between nodes
  - Paths in the underlying undirected graph



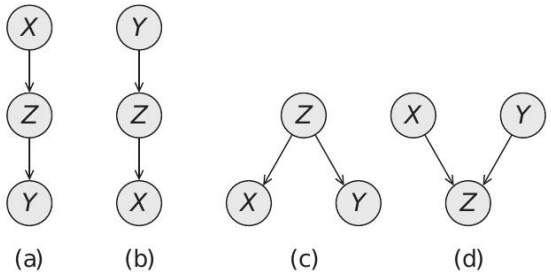
# Conditional independence

- How does dependence "flow" through a network?
- For neighbouring nodes, dependence flows both ways
  - If $x \to y$, knowing $x$ tells us about $y$ and vice versa
- Examine trails between nodes
  - Paths in the underlying undirected graph
- Basic trails — (undirected) paths of length 2
  - Four basic trails

$P(X \perp Y \mid Z)$



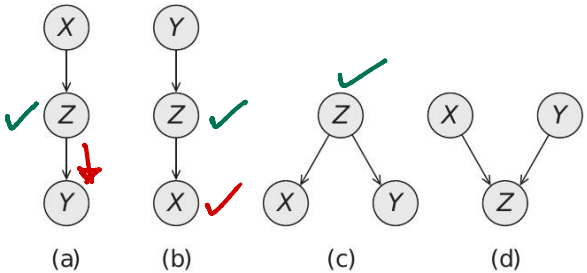
# Basic trails



# Basic trails

- (a), (b) and (c): $Z$ blocks flow between $X$ and $Y$, by semantics of Bayesian networks

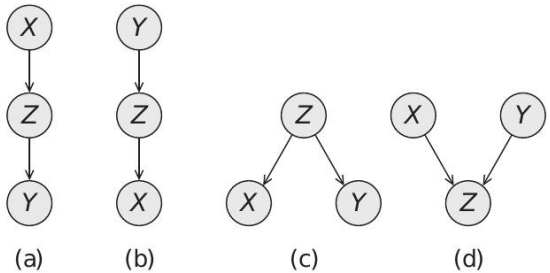
$P(X \perp Y \mid Z)$



# Basic trails

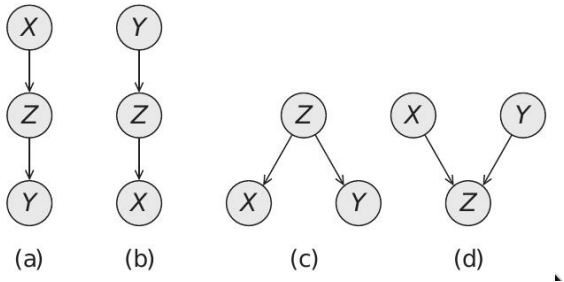
- (a), (b) and (c): $Z$ blocks flow between $X$ and $Y$, by semantics of Bayesian networks
- In (d), knowing $Z$ allows influence to flow

$P(X \perp Y \mid Z)$ ✗



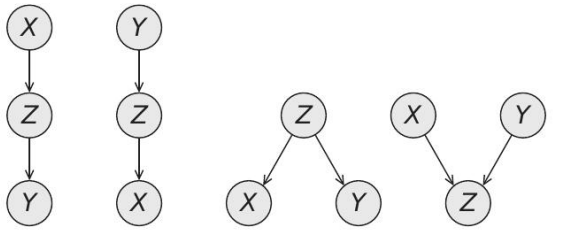
# Basic trails

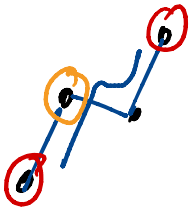
- (a), (b) and (c): $Z$ blocks flow between $X$ and $Y$, by semantics of Bayesian networks
- In (d), knowing $Z$ allows influence to flow
  - $Z$: Car does not start
    $X$: Low Battery, $Y$: No Fuel



# Basic trails

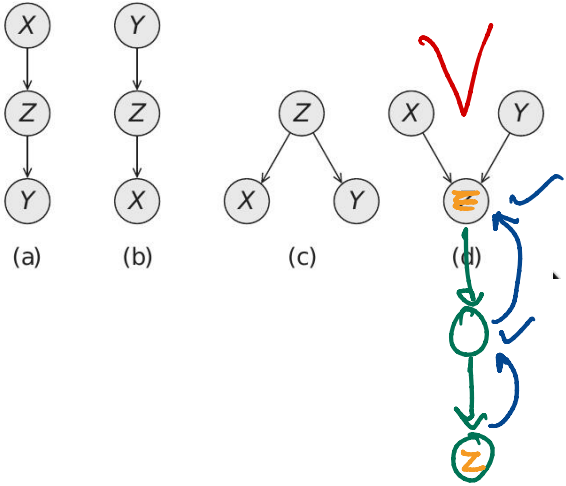
- (a), (b) and (c): $Z$ blocks flow between $X$ and $Y$, by semantics of Bayesian networks
- In (d), knowing $Z$ allows influence to flow
  - $Z$: Car does not start
    $X$: Low Battery, $Y$: No Fuel
  - $Z$: Grass is wet
    $X$: Overnight rain, $Y$: Sprinkler ran



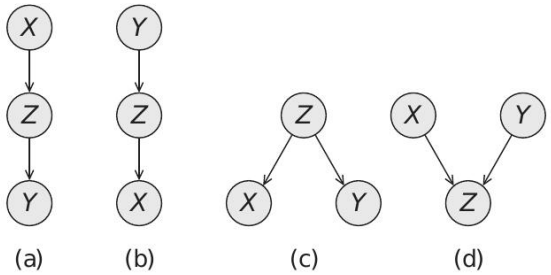
# Basic trails

- (a), (b) and (c): $Z$ blocks flow between $X$ and $Y$, by semantics of Bayesian networks
- In (d), knowing $Z$ allows influence to flow
  - $Z$: Car does not start
    $X$: Low Battery, $Y$: No Fuel
  - $Z$: Grass is wet
    $X$: Overnight rain, $Y$: Sprinkler ran
  - Simplest form of V-structure



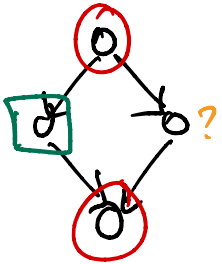
# D-Separation

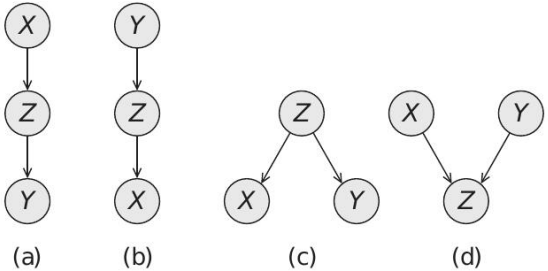
- Check if $X \perp Y \mid Z$



# D-Separation

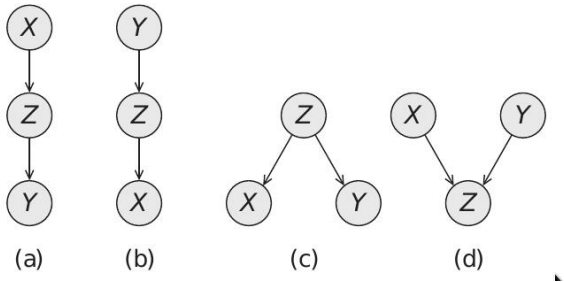
- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$



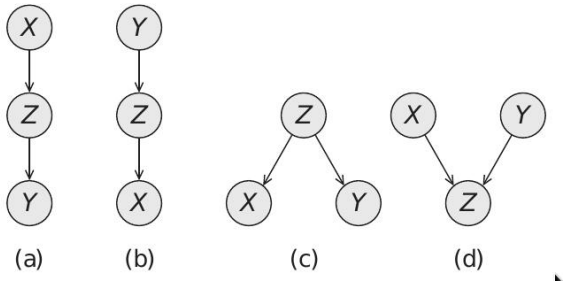
# D-Separation

- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails



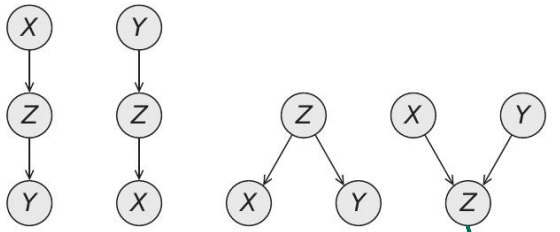
# D-Separation

- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present



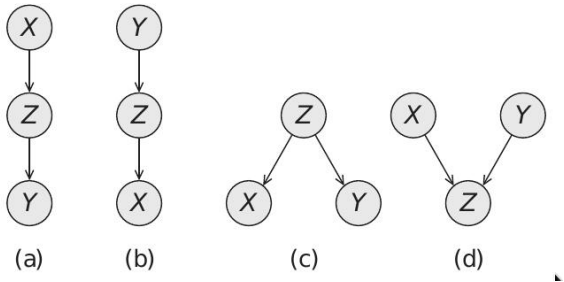
# D-Separation

- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present
  - For (d), need $Z$ absent



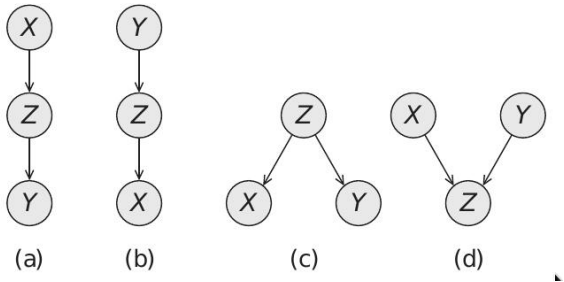
# D-Separation

- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present
  - For (d), need $Z$ absent
  - In general, V-structure includes descendants of the bottom node



# D-Separation

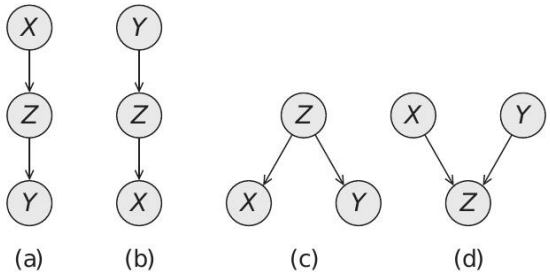
- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present
  - For (d), need $Z$ absent
  - In general, V-structure includes descendants of the bottom node




- $x$ and $y$ are D-separated given $z$ if all trails are blocked

# D-Separation

- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present
  - For (d), need $Z$ absent
  - In general, V-structure includes descendants of the bottom node

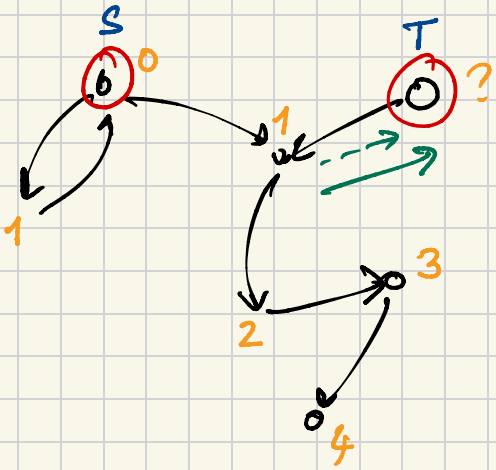



- $x$ and $y$ are D-separated given $z$ if all trails are blocked
- Variation of breadth first search (BFS) to check if $y$ is reachable from $x$ through some trail

Airline network

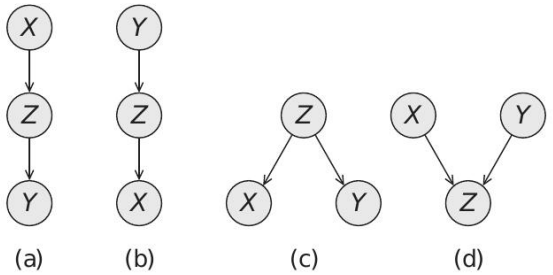
Reachability

Breadth First Search



# D-Separation

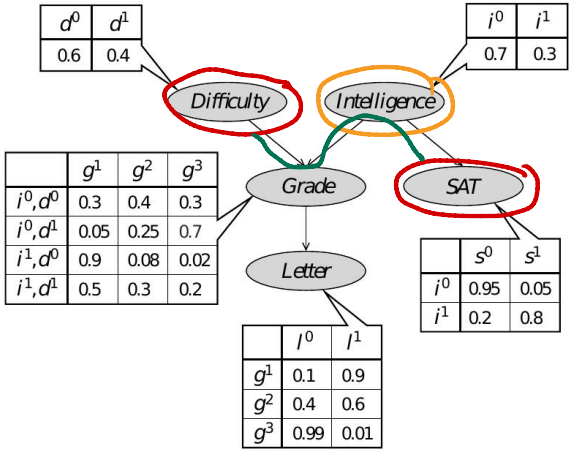
- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from $X$ to $Y$
  - Each undirected path from $X$ to $Y$ is a sequence of basic trails
  - For (a), (b), (c), need $Z$ present
  - For (d), need $Z$ absent
  - In general, V-structure includes descendants of the bottom node




- $x$ and $y$ are D-separated given $z$ if all trails are blocked
- Variation of breadth first search (BFS) to check if $y$ is reachable from $x$ through some trail
- Extends to sets — each $x \in X$ is D-separated from each $y \in Y$

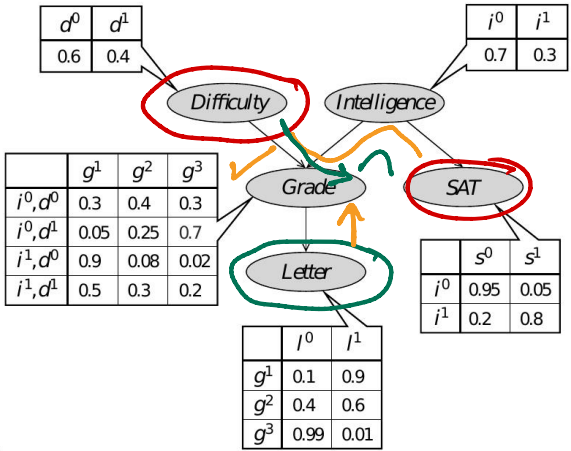
# Conditional independence, example

- Is SAT independent of Difficulty given Intelligence?
  - Yes, Difficulty – Grade – Intelligence – SAT trail is blocked at Grade (V-structure) and Intelligence



# Conditional independence, example

- Is SAT independent of Difficulty given Intelligence?
  - Yes, Difficulty – Grade – Intelligence – SAT trail is blocked at Grade (V-structure) and Intelligence
- Is SAT independent of Difficulty given Letter?
  - No, Difficulty – Grade – Intelligence – SAT trail is open
  - Letter is known, hence something about Grade is known (V-structure)
  - Intelligence is not known




| $d^0$ | $d^1$ |
|---|---|
| 0.6 | 0.4 |

| $i^0$ | $i^1$ |
|---|---|
| 0.7 | 0.3 |

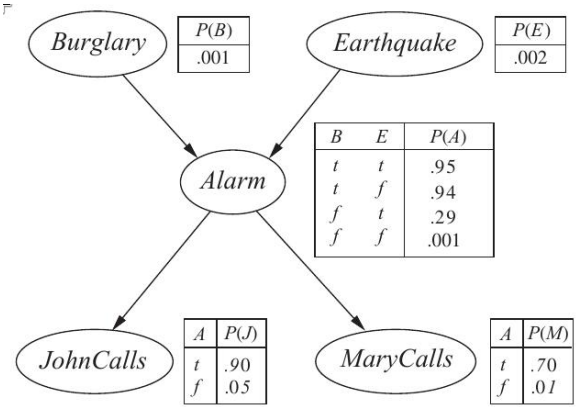
| | $g^1$ | $g^2$ | $g^3$ |
|---|---|---|---|
| $i^0, d^0$ | 0.3 | 0.4 | 0.3 |
| $i^0, d^1$ | 0.05 | 0.25 | 0.7 |
| $i^1, d^0$ | 0.9 | 0.08 | 0.02 |
| $i^1, d^1$ | 0.5 | 0.3 | 0.2 |

| | $s^0$ | $s^1$ |
|---|---|---|
| $i^0$ | 0.95 | 0.05 |
| $i^1$ | 0.2 | 0.8 |

| | $l^0$ | $l^1$ |
|---|---|---|
| $g^1$ | 0.1 | 0.9 |
| $g^2$ | 0.4 | 0.6 |
| $g^3$ | 0.99 | 0.01 |

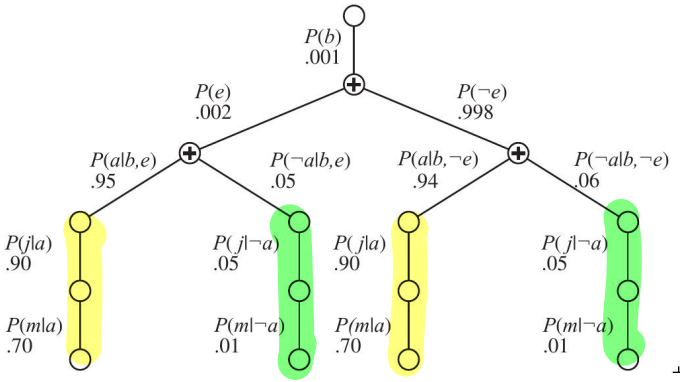
# Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?
- Want $P(b \mid m, j)$
- $\dfrac{P(b, m, j)}{P(m, j)}$
- Use chain rule to evaluate joint probabilities
- Reorder variables appropriately, topological order of graph



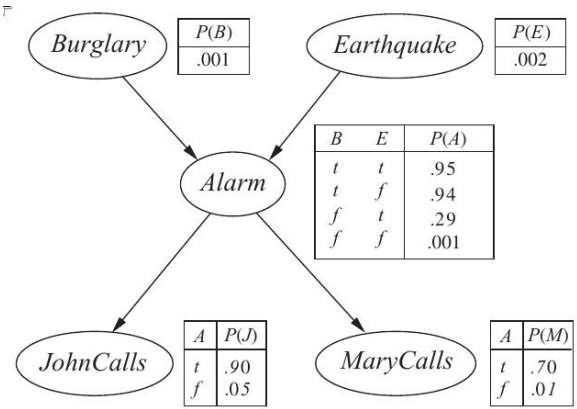
# Computing with probabilistic graphical models

- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$
- Construct the computation tree
- Use dynamic programming to avoid duplicated computations
- However, exact inference is NP-complete, in general
- Instead, approximate inference through sampling



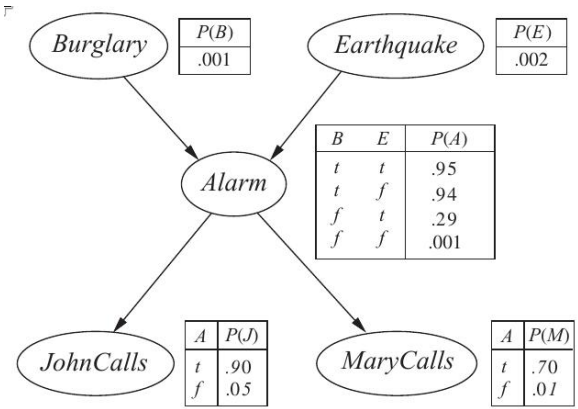
# Approximate inference

- Generate random samples $(b, e, a, m, j)$, count to estimate probabilities



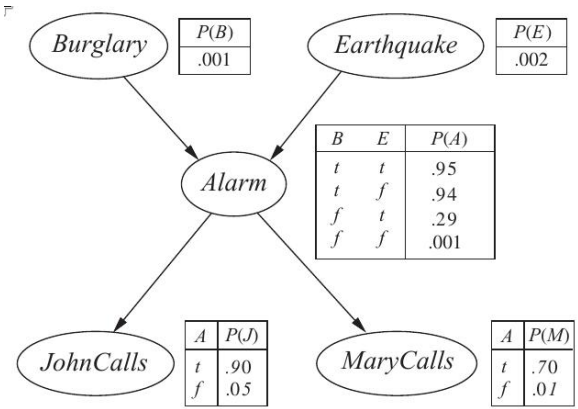
# Approximate inference

- Generate random samples $(b, e, a, m, j)$, count to estimate probabilities
- Random samples should respect conditional probabilities



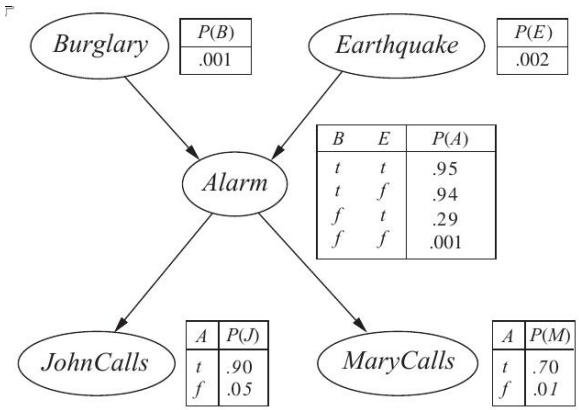
# Approximate inference

- Generate random samples $(b, e, a, m, j)$, count to estimate probabilities
- Random samples should respect conditional probabilities
- Fix parents of $x$ before generating $x$



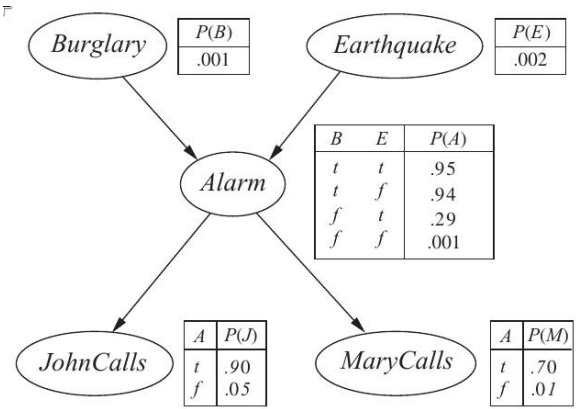
# Approximate inference

- Generate random samples $(b, e, a, m, j)$, count to estimate probabilities
- Random samples should respect conditional probabilities
- Fix parents of $x$ before generating $x$
- Generate in topological order
  - Generate $b$, $e$ with probabilities $P(b)$ and $P(e)$
  - Generate $a$ with probability $P(a \mid b, e)$
  - Generate $j$, $m$ with probabilities $P(j \mid a)$, $P(m \mid a)$



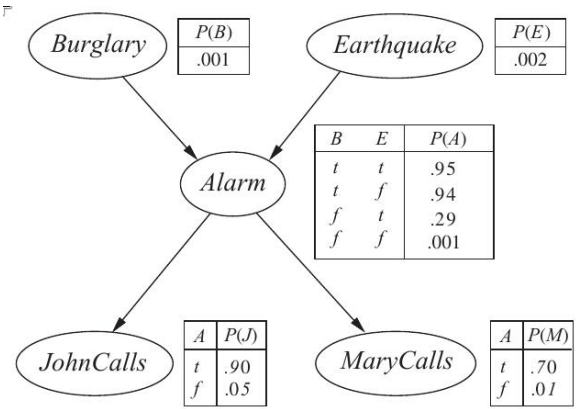
# Approximate inference

- We are interested in $P(b \mid j, m)$



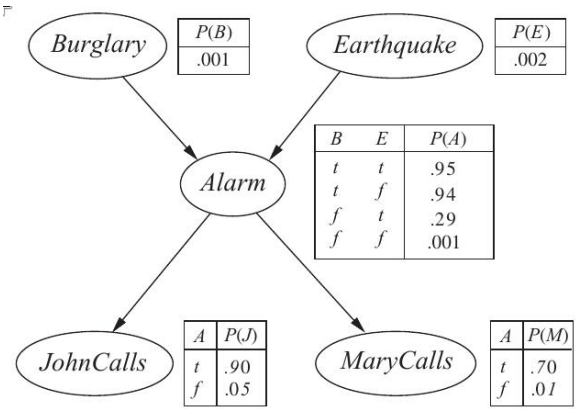
# Approximate inference

- We are interested in $P(b \mid j, m)$
- Samples with $\neg j$ or $\neg m$ are useless



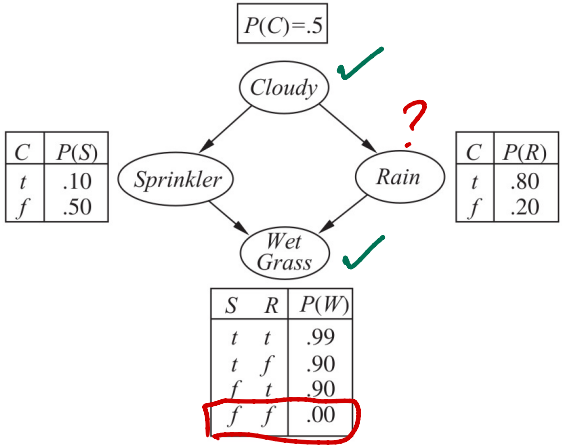
# Approximate inference

- We are interested in $P(b \mid j, m)$
- Samples with $\neg j$ or $\neg m$ are useless
- Can we sample more efficiently?



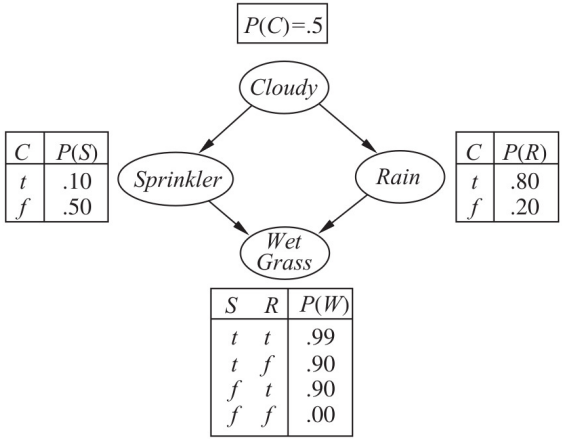
# Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$



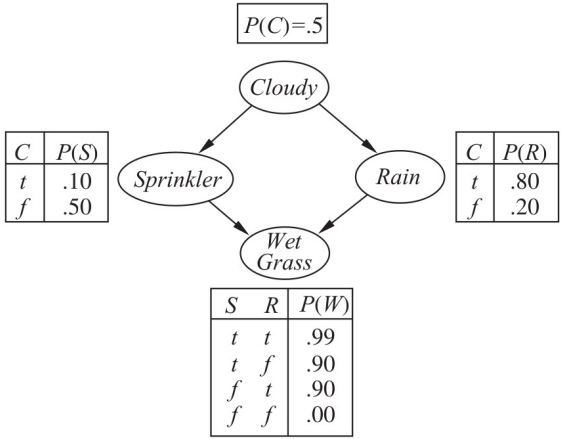
# Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*



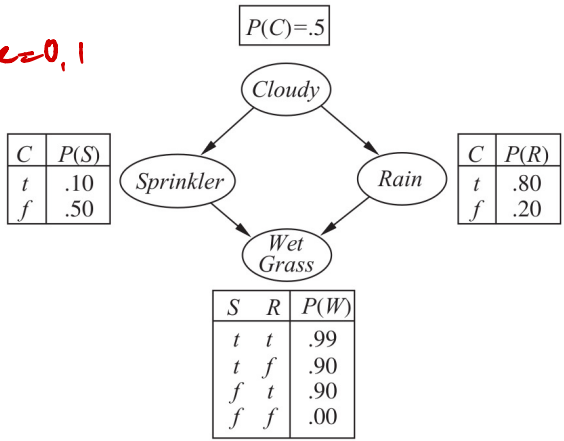
# Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*
- If we start with $\neg Cloudy$, sample is useless



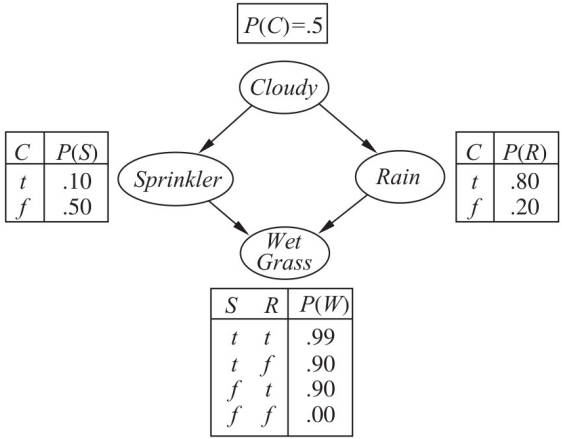
# Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*
- If we start with $\neg Cloudy$, sample is useless
- Immediately stop and reject this sample — rejection sampling



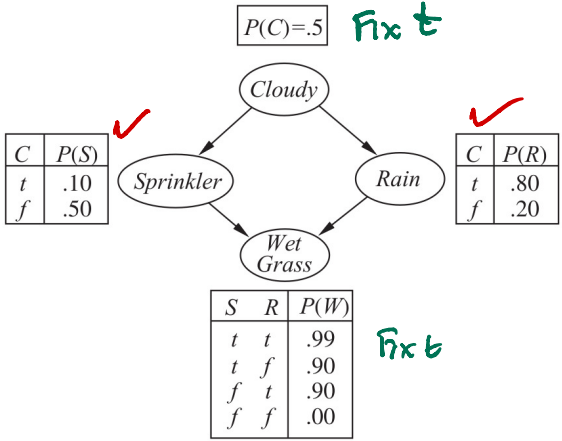
# Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*
- If we start with $\neg Cloudy$, sample is useless
- Immediately stop and reject this sample — rejection sampling
- General problem with low probability situation — many samples are rejected



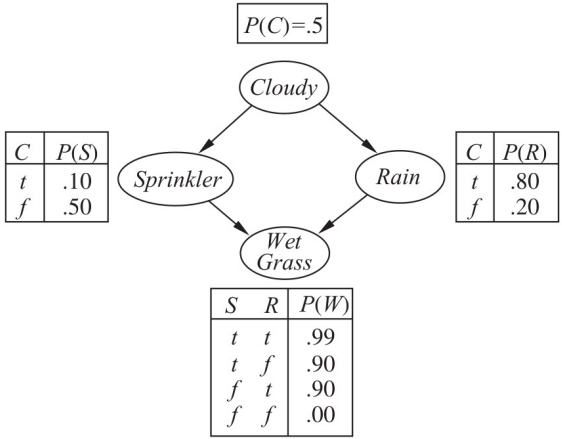
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$



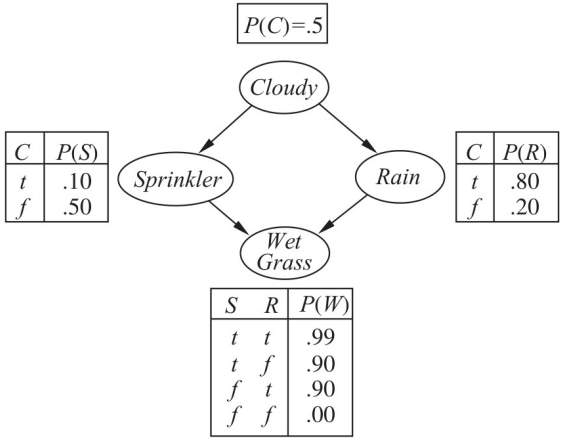
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true



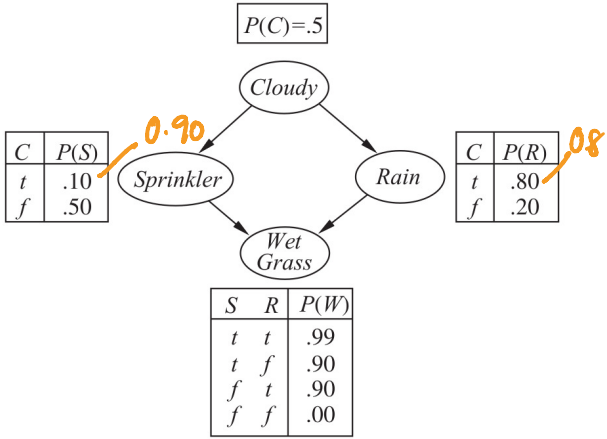
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables



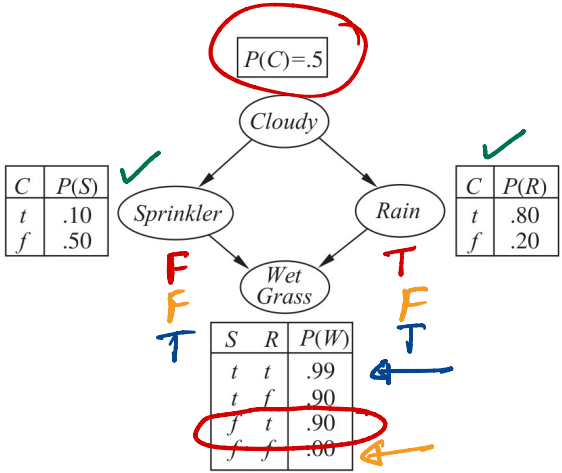
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$



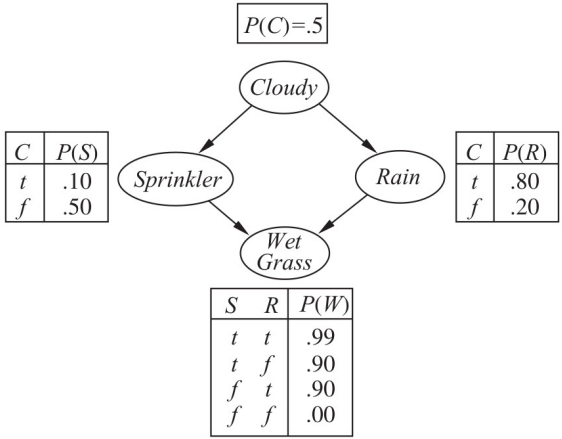
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence:
  $0.5 \times 0.9 = 0.45$



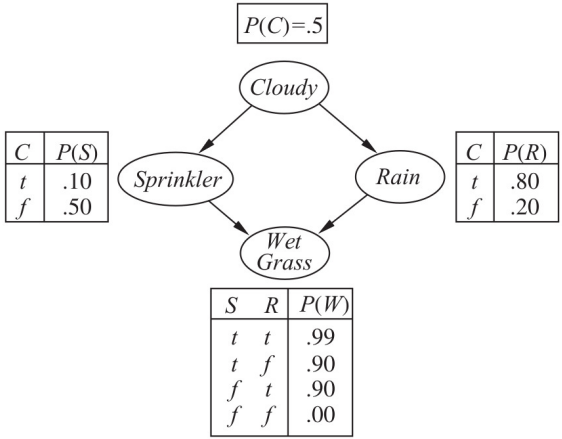
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample



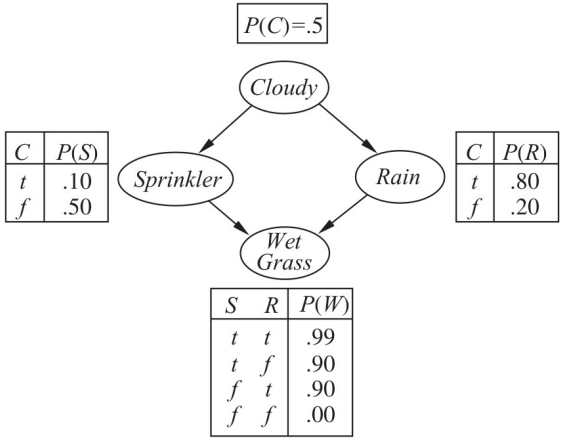
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample
- Samples $s_1, s_2, \ldots, s_N$ with weights $w_1, w_2, \ldots w_N$



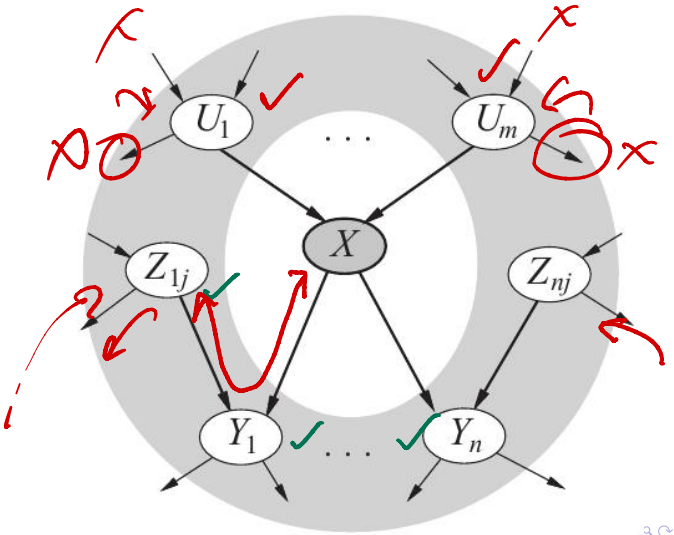
# Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample
- Samples $s_1, s_2, \ldots, s_N$ with weights $w_1, w_2, \ldots w_N$
- $P(r \mid c, w) = \dfrac{\sum_{s_i \text{ has } rain} w_i}{\sum_{1 \le j \le N} w_j}$



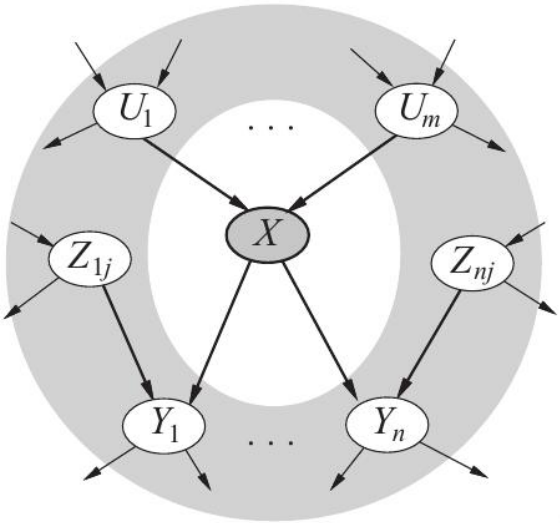
# Markov blanket

- $MB(X)$ — Markov blanket of $X$



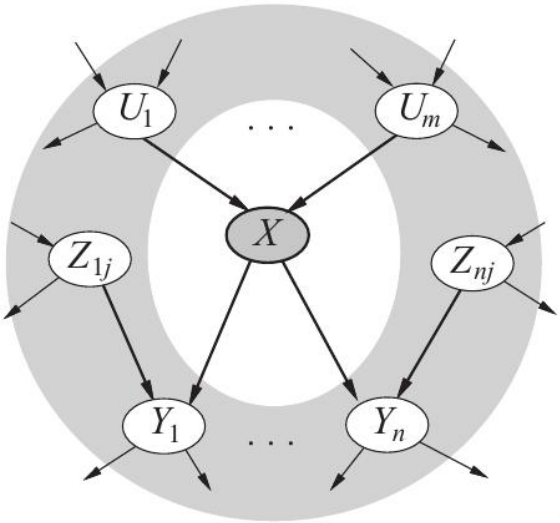
# Markov blanket

- $MB(X)$ — Markov blanket of $X$
  - $Parents(X)$



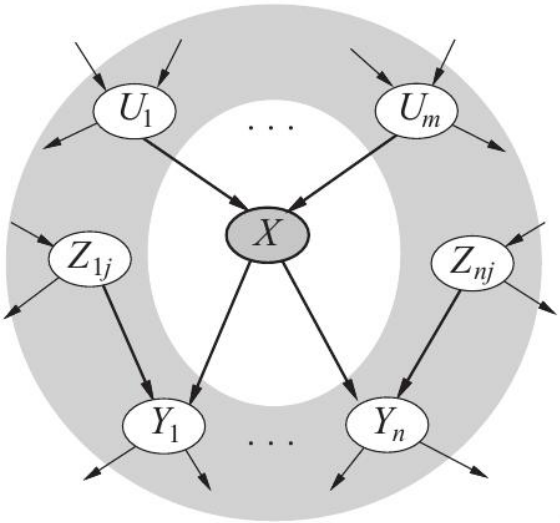
# Markov blanket

- $MB(X)$ — Markov blanket of $X$
  - $Parents(X)$
  - $Children(X)$



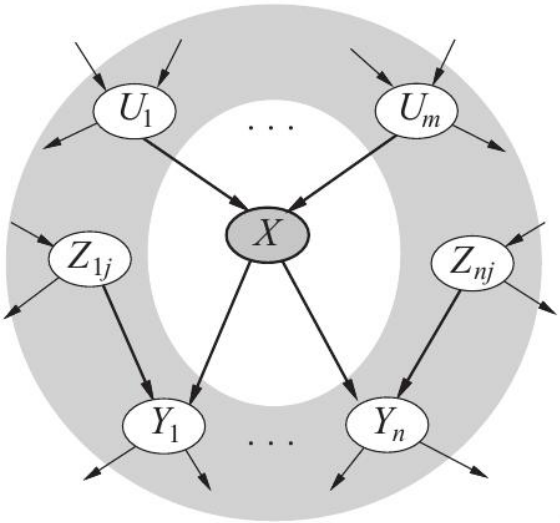
# Markov blanket

- $MB(X)$ — Markov blanket of $X$
  - $Parents(X)$
  - $Children(X)$
  - $Parents\ of\ Children(X)$



# Markov blanket

- $MB(X)$ — Markov blanket of $X$
  - $Parents(X)$
  - $Children(X)$
  - $Parents\ of\ Children(X)$
- $X \perp \neg MB(X) \mid MB(X)$



# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$

# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$
- Move probabilistically from $s_j = (x_1, x_2, \ldots, x_n)$ to $s_k = (y_1, y_2, \ldots, y_n)$

# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$
- Move probabilistically from $s_j = (x_1, x_2, \ldots, x_n)$ to $s_k = (y_1, y_2, \ldots, y_n)$
- Allow such a move only when $s_j$, $s_k$ differ at exactly one position
  - $s_j = (x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$
  - $s_k = (x_1, x_2, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$

# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$
- Move probabilistically from $s_j = (x_1, x_2, \ldots, x_n)$ to $s_k = (y_1, y_2, \ldots, y_n)$
- Allow such a move only when $s_j$, $s_k$ differ at exactly one position
  - $s_j = (x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$
  - $s_k = (x_1, x_2, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$
- Sampling algorithm
  - Current state is $s_j = (x_1, x_2, \ldots, x_n)$
  - Choose $i$ uniformly in $[1, n]$
  - Resample $x_i$ given current values $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
  - Random walk through state space — count number of visits to each state

# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$
- Move probabilistically from $s_j = (x_1, x_2, \ldots, x_n)$ to $s_k = (y_1, y_2, \ldots, y_n)$
- Allow such a move only when $s_j$, $s_k$ differ at exactly one position
  - $s_j = (x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$
  - $s_k = (x_1, x_2, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$
- Sampling algorithm
  - Current state is $s_j = (x_1, x_2, \ldots, x_n)$
  - Choose $i$ uniformly in $[1, n]$
  - Resample $x_i$ given current values $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
  - Random walk through state space — count number of visits to each state
- Need to compute $P[y_i \mid x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$

# Gibbs sampling

- State of a Bayesian network is a valuation of variables $(V_1, V_2, \ldots, V_n)$
- Move probabilistically from $s_j = (x_1, x_2, \ldots, x_n)$ to $s_k = (y_1, y_2, \ldots, y_n)$
- Allow such a move only when $s_j$, $s_k$ differ at exactly one position
  - $s_j = (x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$
  - $s_k = (x_1, x_2, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$
- Sampling algorithm
  - Current state is $s_j = (x_1, x_2, \ldots, x_n)$
  - Choose $i$ uniformly in $[1, n]$
  - Resample $x_i$ given current values $(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
  - Random walk through state space — count number of visits to each state
- Need to compute $P[y_i \mid x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$
- Why does this work at all?

# Approximate inference using Markov chains

## Markov chains

- Finite set of states, with transition probabilities between states

# Approximate inference using Markov chains

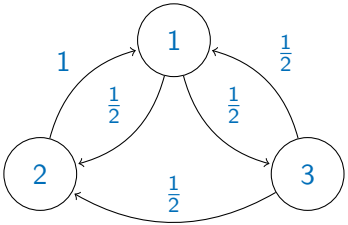
## Markov chains

- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables

# Approximate inference using Markov chains

## Markov chains

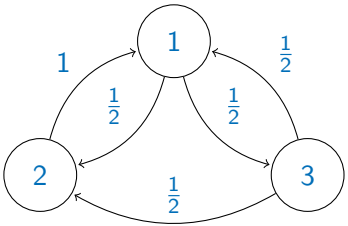
- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables
- A three state Markov Chain



# Approximate inference using Markov chains

## Markov chains

- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables
- A three state Markov Chain



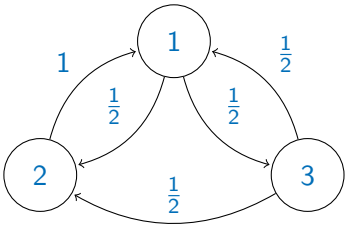

- Represent using a transition matrix — stochastic

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

# Approximate inference using Markov chains

## Markov chains

- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables
- A three state Markov Chain




- Represent using a transition matrix — stochastic

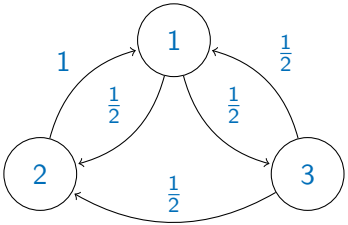
$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

- $P[j]$ is probability of being in state $j$

# Approximate inference using Markov chains

## Markov chains

- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables
- A three state Markov Chain




- Represent using a transition matrix — stochastic

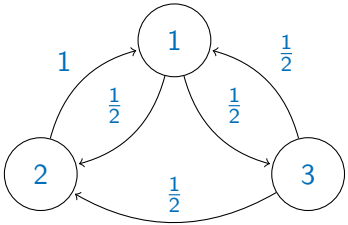
$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

- $P[j]$ is probability of being in state $j$
- Start in state 1, so initially $P = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

# Markov chains . . .

- After one step:

$$P^\top A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$



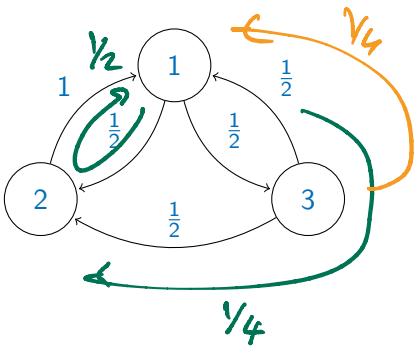
# Markov chains . . .

- After one step:

$$P^\top A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

- After second step:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$



# Markov chains ...

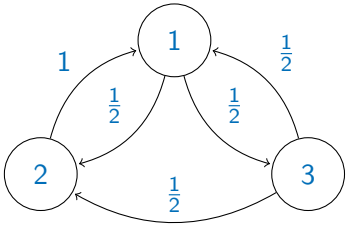
- After one step:

$$P^\top A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

- After second step:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

- After $k$ steps, $P[j]$ is probability of being in state $j$



# Markov chains . . .

- After one step:

$$P^\top A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

- After second step:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

- After $k$ steps, $P[j]$ is probability of being in state $j$
- Continuing our example,

$$\begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix} \to \begin{bmatrix} \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{bmatrix} \to \begin{bmatrix} \frac{9}{16} & \frac{5}{16} & \frac{1}{8} \end{bmatrix}$$