Lecture 7
Inference in Bayesian Networks

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Slides by Stuart Russell and Peter Norvig

Course Overview

- ✔ Introduction
  - ✔ Artificial Intelligence
  - ✔ Intelligent Agents
- ✔ Search
  - ✔ Uninformed Search
  - ✔ Heuristic Search
- Uncertain knowledge and Reasoning
  - ✔ Probability and Bayesian approach
  - Bayesian Networks
  - Hidden Markov Chains
  - Kalman Filters
- Learning
  - Supervised
    Learning Bayesian Networks, Neural Networks
  - Unsupervised
    EM Algorithm
- Reinforcement Learning
- Games and Adversarial Search
  - Minimax search and Alpha-beta pruning
  - Multiagent search
- Knowledge representation and Reasoning
  - Propositional logic
  - First order logic
  - Inference
  - Plannning

Bayesian networks, Resume

Encode local conditional independences
$\Pr(X_i \mid X_{-i}) = \Pr(X_i \mid Parents(X_i))$

Thus the global semantics simplifies to (joint probability factorization):

$$\Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i \mid X_1, \ldots, X_{i-1}) \quad \text{(chain rule)}$$

$$= \prod_{i=1}^{n} \Pr(X_i \mid Parents(X_i)) \quad \text{(by construction)}$$

Outline

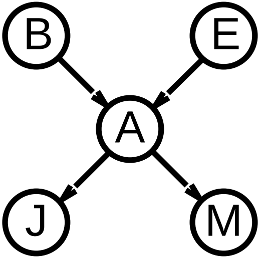
Inference tasks

- Simple queries: compute posterior marginal $\Pr(X_i \mid \mathbf{E} = \mathbf{e})$
  e.g., $P(NoGas \mid Gauge = empty, Lights = on, Starts = false)$
- Conjunctive queries:
  $\Pr(X_i, X_j \mid \mathbf{E} = \mathbf{e}) = \Pr(X_i \mid \mathbf{E} = \mathbf{e}) \Pr(X_j \mid X_i, \mathbf{E} = \mathbf{e})$
- Optimal decisions: decision networks include utility information;
  probabilistic inference required for $P(outcome \mid action, evidence)$
- Value of information: which evidence to seek next?
- Sensitivity analysis: which probability values are most critical?
- Explanation: why do I need a new starter motor?

Inference by enumeration

Sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:



$$\begin{aligned}\Pr(B \mid j, m) &= \Pr(B, j, m)/P(j, m) \\ &= \alpha \Pr(B, j, m) \\ &= \alpha \sum_e \sum_a \Pr(B, e, a, j, m)\end{aligned}$$

Rewrite full joint entries using product of CPT entries:

$$\begin{aligned}\Pr(B \mid j, m) &= \alpha \sum_e \sum_a \Pr(B) P(e) \Pr(a \mid B, e) P(j \mid a) P(m \mid a) \\ &= \alpha \Pr(B) \sum_e P(e) \sum_a \Pr(a \mid B, e) P(j \mid a) P(m \mid a)\end{aligned}$$

Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

Enumeration algorithm

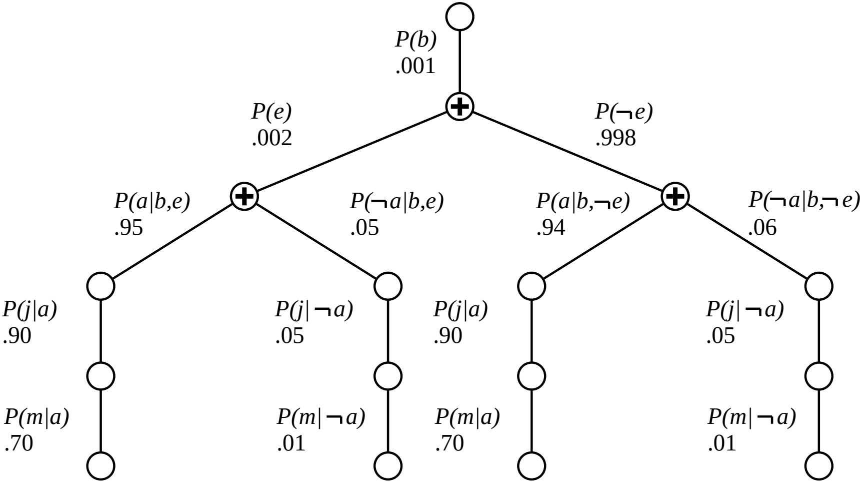
function Enumeration-Ask(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    $\mathbf{Q}(x_i) \leftarrow$ Enumerate-All(bn.Vars, $\mathbf{e} \cup \{X = x_i\}$)
  return Normalize($\mathbf{Q}(X)$)

---

function Enumerate-All($vars$, $\mathbf{e}$) **returns** a real number
  if Empty?($vars$) **then return** 1.0
  $Y \leftarrow$ First($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid parent(Y)) \times$ Enumerate-All(Rest($vars$), $\mathbf{e}$)
  else return $\sum_y P(y \mid parent(Y)) \times$ Enumerate-All(Rest($vars$), $\mathbf{e} \cup \{Y = y\}$)

Evaluation tree



Enumeration is inefficient: repeated computation
  e.g., computes $P(j \mid a)P(m \mid a)$ for each value of e

Inference by variable elimination

Variable elimination: carry out summations right-to-left,
storing intermediate results (factors) to avoid recomputation

$$
\begin{aligned}
&\Pr(B \mid j, m) \\
&= \alpha \underbrace{\Pr(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum_a \underbrace{\Pr(a \mid B, e)}_{A} \underbrace{P(j \mid a)}_{J} \underbrace{P(m \mid a)}_{M} \\
&= \alpha \Pr(B) \sum_e P(e) \sum_a \Pr(a \mid B, e) P(j \mid a) f_M(a) \\
&= \alpha \Pr(B) \sum_e P(e) \sum_a \Pr(a \mid B, e) f_J(a) f_M(a) \\
&= \alpha \Pr(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
&= \alpha \Pr(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \text{ (sum out } A\text{)} \\
&= \alpha \Pr(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)} \\
&= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

Variable elimination: Basic operations

Summing out a variable from a product of factors:

1. move any constant factors outside the summation:

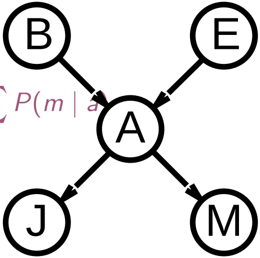
   $\sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k =$
   $f_1 \times \cdots \times f_i \times f_{\bar{X}}$
   assuming $f_1, \ldots, f_i$ do not depend on X

2. add up submatrices in pointwise product of remaining factors:

   Eg: pointwise product of f_1 and f_2:
   $f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)$
   $= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)$
   E.g., $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Irrelevant variables

Consider the query $P(JohnCalls \mid Burglary = true)$



$$P(J \mid b) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(J \mid a) \sum_m P(m \mid a)$$

Sum over m is identically 1; M is **irrelevant** to the query

Theorem

Y is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathbf{E})$

Here, $X = JohnCalls$, $\mathbf{E} = \{Burglary\}$, and $Ancestors(\{X\} \cup \mathbf{E}) = \{Alarm, Earthquake\}$ so $MaryCalls$ is irrelevant

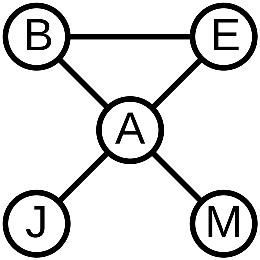
Irrelevant variables contd.

Defn: moral graph of DAG Bayes net: marry all parents and drop arrows
Defn: **A** is m-separated from **B** by **C** iff separated by **C** in the moral graph

Theorem

Y is irrelevant if m-separated from X by E

For $P(JohnCalls \mid Alarm = true)$, both *Burglary* and *Earthquake* are irrelevant



Complexity of exact inference

Singly connected networks (or polytrees):
- any two nodes are connected by at most one (undirected) path
- time and space cost (with variable elimination) are $O(d^k n)$
- hence time and space cost are linear in n and k bounded by a constant

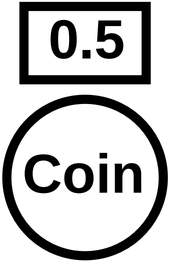
Multiply connected networks:
- can reduce 3SAT to exact inference $\Longrightarrow$ NP-hard
- equivalent to **counting** 3SAT models $\Longrightarrow$ #P-complete

Proof of this in one of the exercises for Thursday.

Inference by stochastic simulation

Basic idea:

- Draw N samples from a sampling distribution S
- Compute an approximate posterior probability $\hat{P}$
- Show this converges to the true probability P



Outline:

– Sampling from an empty network
– Rejection sampling: reject samples disagreeing with evidence
– Likelihood weighting: use evidence to weight samples
– Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

Sampling from an empty network

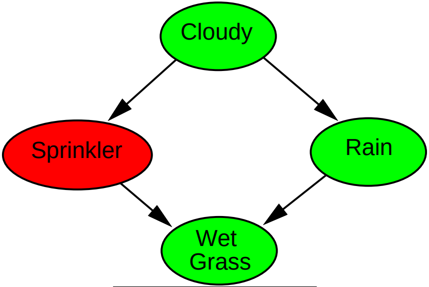

```
function Prior-Sample(bn) returns an event sampled from bn
  inputs: bn, a belief network specifying joint distribution Pr(X_1, ..., X_n)

   x ← an event with n elements
   for i = 1 to n do
        x_i ← a random sample from Pr(X_i | parents(X_i))
             given the values of Parents(X_i) in x
   return x
```

Ancestor sampling

Example

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an empty network contd.

Probability that PriorSample generates a particular event

$$S_{PS}(x_1 \ldots x_n) = P(x_1 \ldots x_n)$$

i.e., the true prior probability

E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Proof: Let $N_{PS}(x_1 \ldots x_n)$ be the number of samples generated for event $x_1, \ldots, x_n$. Then we have

$$\begin{aligned} \lim_{N \to \infty} \hat{P}(x_1, \ldots, x_n) &= \lim_{N \to \infty} N_{PS}(x_1, \ldots, x_n)/N \\ &= S_{PS}(x_1, \ldots, x_n) \\ &= \prod_{i=1}^{n} P(x_i | parents(X_i)) = P(x_1 \ldots x_n) \end{aligned}$$

$\leadsto$That is, estimates derived from PriorSample are consistent
Shorthand: $\hat{P}(x_1, \ldots, x_n) \approx P(x_1 \ldots x_n)$

Rejection sampling

$\hat{\Pr}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

```
function Rejection-Sampling(X, e, bn, N) returns an estimate of P(X|e)
   local variables: N, a vector of counts over X, initially zero

   for j = 1 to N do
       x ← Prior-Sample(bn)
       if x is consistent with e then
           N[x] ← N[x]+1 where x is the value of X in x
   return Normalize(N[X])
```

E.g., estimate $\Pr(Rain|Sprinkler = true)$ using 100 samples
27 samples have $Sprinkler = true$
Of these, 8 have $Rain = true$ and 19 have $Rain = false$.

$\hat{\Pr}(Rain|Sprinkler = true) = \text{Normalize}(\langle 8, 19\rangle) = \langle 0.296, 0.704\rangle$
Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

Rejection sampling returns consistent posterior estimates

Proof:
$\hat{\Pr}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e})$ (algorithm defn.)
$= \mathbf{N}_{PS}(X, \mathbf{e}) / N_{PS}(\mathbf{e})$ (normalized by $N_{PS}(\mathbf{e})$)
$\approx \Pr(X, \mathbf{e}) / P(\mathbf{e})$ (property of PriorSample)
$= \Pr(X|\mathbf{e})$ (defn. of conditional probability)

Problem: hopelessly expensive if $P(\mathbf{e})$ is small
$P(\mathbf{e})$ drops off exponentially with number of evidence variables!

Likelihood weighting

Idea: fix evidence variables, sample only nonevidence variables,
and weight each sample by the likelihood it accords the evidence

```
function Likelihood-Weighting(X, e, bn, N) returns an estimate of P(X|e)
    local variables: W, a vector of weighted counts over X, initially zero

    for j = 1 to N do
        x, w ← Weighted-Sample(bn)
        W[x] ← W[x] + w where x is the value of X in x
    return Normalize(W[X])
```

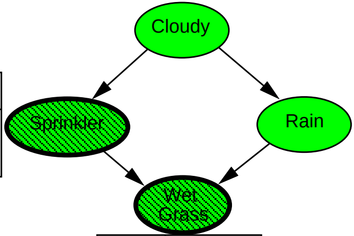
```
function Weighted-Sample(bn, e) returns an event and a weight

    x ← an event with n elements; w ← 1
    for i = 1 to n do
        if X_i has a value x_i in e
            then w ← w × P(X_i = x_i | parents(X_i))
            else x_i ← a random sample from Pr(X_i | parents(X_i))
    return x, w
```

Likelihood weighting example

$P(Rain|Sprinkler = true, WetGrass = true)$

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

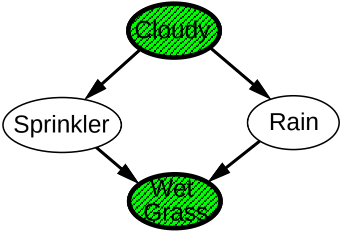
Likelihood weighting analysis

Likelihood weighting returns consistent estimates

Sampling probability for WeightedSample is

$$S_{WS}(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i|parents(Z_i))$$

(pays attention to evidence in **ancestors** only)
$\leadsto$somewhere "in between" prior and posterior distribution



Weight for a given sample $\mathbf{z},\mathbf{e}$ is

$$w(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{m} P(e_i|parents(E_i))$$

but performance still degrades with many evidence variables because a few samples have nearly all the total weight

Weighted sampling probability is

$$S_{WS}(\mathbf{z},\mathbf{e})w(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i|parents(Z_i)) \prod_{i=1}^{m} P(e_i|parents(E_i)) = P(\mathbf{z},\mathbf{e})$$

Summary

Approximate inference by LW:

- LW does poorly when there is lots of (late-in-the-order) evidence
- LW generally insensitive to topology
- Convergence can be very slow with probabilities close to 1 or 0
- Can handle arbitrary combinations of discrete and continuous variables

Approximate inference using MCMC

"State" of network = current assignment to all variables.
Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

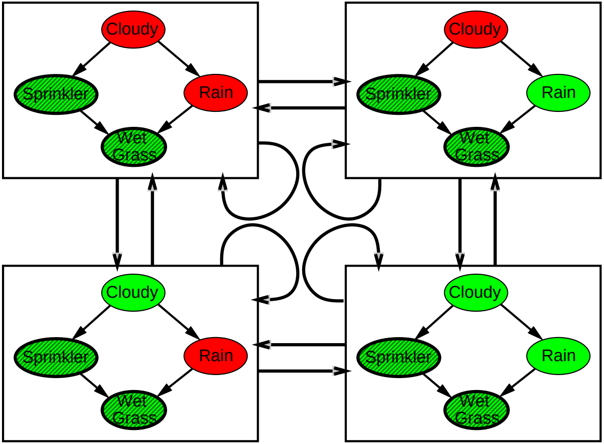
function MCMC-Ask($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
    local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
        $\mathbf{Z}$, nonevidence variables in bn, hidden + query
        $\mathbf{x}$, current state of the network, initially copied from $\mathbf{e}$

    initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
    for $j = 1$ to N **do**
        $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
        for each Z_i in $\mathbf{Z}$ **do**
            sample the value of Z_i in $\mathbf{x}$ from $\Pr(Z_i|mb(Z_i))$
                given the values of $MB(Z_i)$ in $\mathbf{x}$
    return Normalize($\mathbf{N}[X]$)

Can also choose a variable to sample at random each time

The Markov chain

With $Sprinkler = true, WetGrass = true$, there are four states:



Wander about for a while, average what you see

Probabilistic finite state machine

MCMC example contd.

Estimate $\Pr(Rain|Sprinkler = true, WetGrass = true)$

Sample *Cloudy* or *Rain* given its Markov blanket, repeat.
Count number of times *Rain* is true and false in the samples.

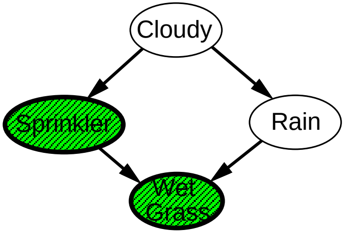
E.g., visit 100 states
31 have $Rain = true$, 69 have $Rain = false$

$\hat{\Pr}(Rain|Sprinkler = true, WetGrass = true) = \text{Normalize}(\langle 31, 69\rangle) = \langle 0.31, 0.69\rangle$

Theorem

The Markov Chain approaches a stationary distribution: long-run fraction of time spent in each state is exactly proportional to its posterior probability

Markov blanket sampling



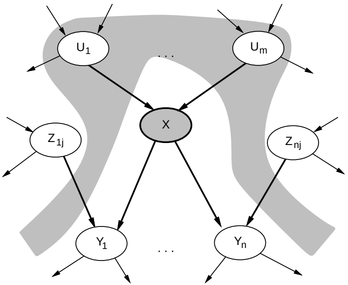
Markov blanket of *Cloudy* is
Sprinkler and *Rain*

Markov blanket of *Rain* is
Cloudy, *Sprinkler*, and *WetGrass*

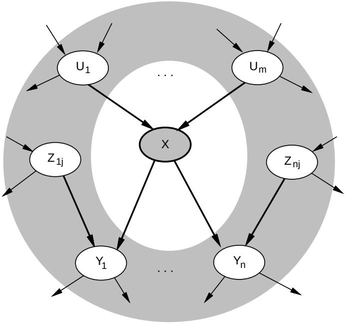
Main computational problems:
1) Difficult to tell if convergence has been achieved
2) Can be wasteful if Markov blanket is large:
$P(X_i|mb(X_i))$ won't change much (law of large numbers)

Local semantics and Markov Blanket

Local semantics: each node is conditionally independent of its nondescendants given its parents



Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



MCMC analysis: Outline

- Transition probability $q(\mathbf{x} \rightarrow \mathbf{x}')$
- Occupancy probability $\pi_t(\mathbf{x})$ at time t
- Equilibrium condition on π_t defines stationary distribution $\pi(\mathbf{x})$
  Note: stationary distribution depends on choice of $q(\mathbf{x} \rightarrow \mathbf{x}')$
- Pairwise detailed balance on states guarantees equilibrium
- Gibbs sampling transition probability:
  sample each variable given current values of all others
  $\Longrightarrow$ detailed balance with the true posterior
- For Bayesian networks, Gibbs sampling reduces to sampling conditioned on each variable's Markov blanket

Stationary distribution

- $\pi_t(\mathbf{x})$ = probability in state $\mathbf{x}$ at time t
  $\pi_{t+1}(\mathbf{x}')$ = probability in state $\mathbf{x}'$ at time $t+1$

- π_{t+1} in terms of π_t and $q(\mathbf{x} \rightarrow \mathbf{x}')$

$$\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}')$$

- Stationary distribution: $\pi_t = \pi_{t+1} = \pi$

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') \qquad \text{for all } \mathbf{x}'$$

- If π exists, it is unique (specific to $q(\mathbf{x} \rightarrow \mathbf{x}')$)

- In equilibrium, expected "outflow" = expected "inflow"

Detailed balance

- "Outflow" = "inflow" for each pair of states:

$$\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \qquad \text{for all } \mathbf{x},\ \mathbf{x}'$$

- Detailed balance $\implies$ stationarity:

$$\begin{aligned}
\sum_{\mathbf{x}} \pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') &= \sum_{\mathbf{x}} \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \\
&= \pi(\mathbf{x}') \sum_{\mathbf{x}} q(\mathbf{x}' \to \mathbf{x}) \\
&= \pi(\mathbf{x}')
\end{aligned}$$

- MCMC algorithms typically constructed by designing a transition probability q that is in detailed balance with desired π

Gibbs sampling

- Sample each variable in turn, given **all other variables**
- Sampling X_i, let $\bar{\mathbf{X}}_i$ be all other nonevidence variables
- Current values are x_i and $\bar{\mathbf{x}}_i$; $\mathbf{e}$ is fixed
- Transition probability is given by

$$q(\mathbf{x} \rightarrow \mathbf{x}') = q(x_i, \bar{\mathbf{x}}_i \rightarrow x_i', \bar{\mathbf{x}}_i) = P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e})$$

- This gives detailed balance with true posterior $P(\mathbf{x}|\mathbf{e})$:

$$\begin{aligned}
\pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') &= P(\mathbf{x}|\mathbf{e}) P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e}) = P(x_i, \bar{\mathbf{x}}_i | \mathbf{e}) P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e}) \\
&= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e}) P(\bar{\mathbf{x}}_i | \mathbf{e}) P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e}) \quad \text{(chain rule)} \\
&= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e}) P(x_i', \bar{\mathbf{x}}_i | \mathbf{e}) \qquad \text{(chain rule backwards)} \\
&= q(\mathbf{x}' \rightarrow \mathbf{x}) \pi(\mathbf{x}') = \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x})
\end{aligned}$$

Summary

Exact inference by variable elimination:
- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by LW, MCMC:

- PriorSampling and RejectionSampling unusable as evidence grow
  - LW does poorly when there is lots of (late-in-the-order) evidence
  - LW, MCMC generally insensitive to topology
  - Convergence can be very slow with probabilities close to 1 or 0
  - Can handle arbitrary combinations of discrete and continuous variables