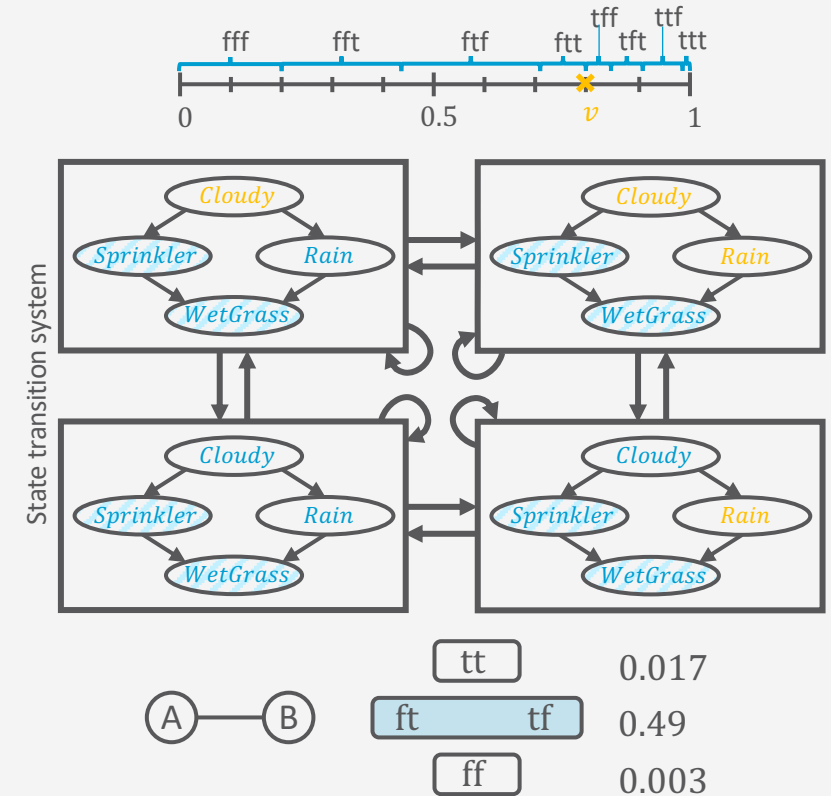


Inference: Approximate Probabilistic Inference

Statistical Relational Artificial Intelligence
(StaRAI)



Contents

1. Introduction

- Notions of AI, agents, StaRAI

2. Modelling Formalisms

- a. *Logic*
- b. *Probabilistic Graphical Models (PGMs)*
- c. *Probabilistic Relational Models (PRMs)*
- Syntax, semantics, inference tasks

3. Inference

- a. *Exact Inference*
 - i. *(Weighted) Model Counting*
 - ii. *Variable Elimination*
 - iii. *Lifted Inference*

b. *Approximate Inference*

- *Inference algorithms*

4. Learning

- Relation learning in logic
- Parameter / structure learning for PGMs
- From PGMs to PRMs, PRM learning

5. Sequential Models and Inference

- Syntax, semantics, inference tasks, algorithms

6. Decision-theoretic Models and Inference

- Preferences, utility
- Syntax, semantics, inference task, algorithm

Outline: 3. Inference

a. Exact Inference

i. (Weighted) Model Counting

- Model counting
- Weighted model counting, knowledge compilation (KC)

ii. Variable Elimination and the Like

- Variable elimination (VE)
- Junction tree algorithm (JT)

iii. Lifted Inference

- First-order KC, Lifted VE, lifted JT

b. Approximate Inference

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling

Approximations

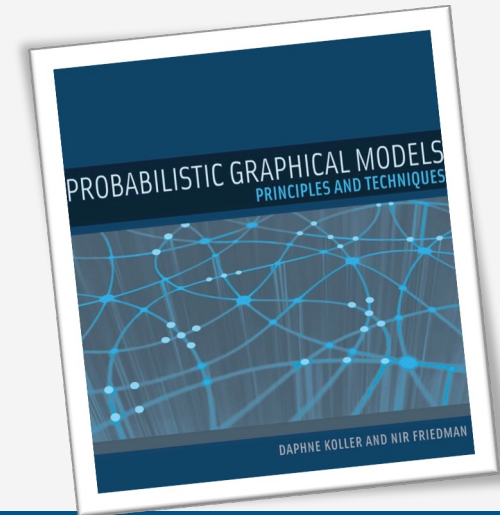
- Approximate answers to queries such as the posterior $P(R|e)$
 - Goal: Overcome complexity of exact inference ($\mathcal{NP}$ -hard problem)
 - However: Problem of approximate inference also $\mathcal{NP}$ -hard in worst case
- Assume an intuitive of approximation: *The answer may be erroneous up to some amount*
- Formally treated in

PAC theory (Probably Approximately Correct)

by parameters (δ, ε)

- Confidence (quantified by δ) in that found solution maximally deviates from true solution up to ε
 - Alternative: How many samples do you need to satisfy δ, ε ?

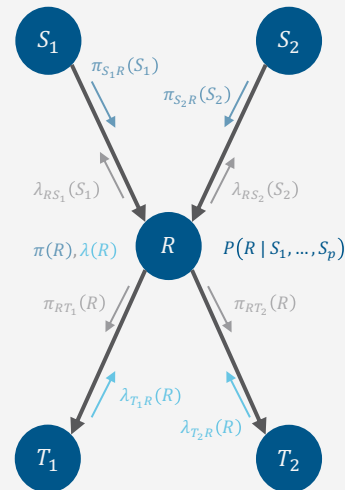
Based on Chapter 12.2, in “Probabilistic Graphical Models”
by Koller & Friedman (2009), also includes information about PAC learning



Deterministic vs. Stochastic Approximations

Deterministic Approximations

- Yields same result in different runs
 - E.g., message passing on cyclic graphs
 - Always follows same schema, computations will be identical in each run, meaning result will be identical
- Known as **loopy belief propagation**



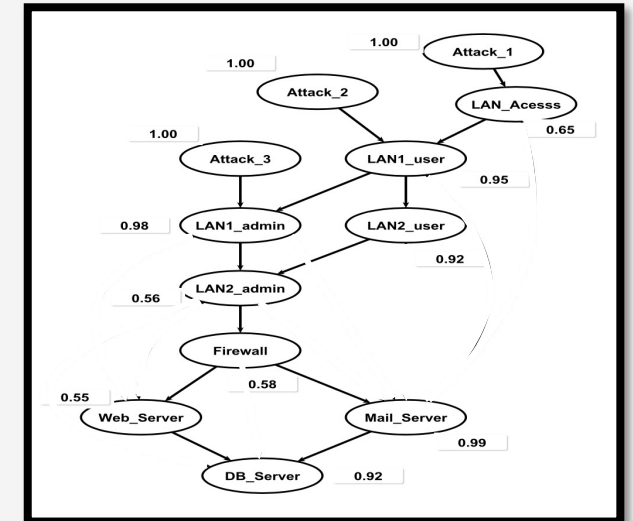
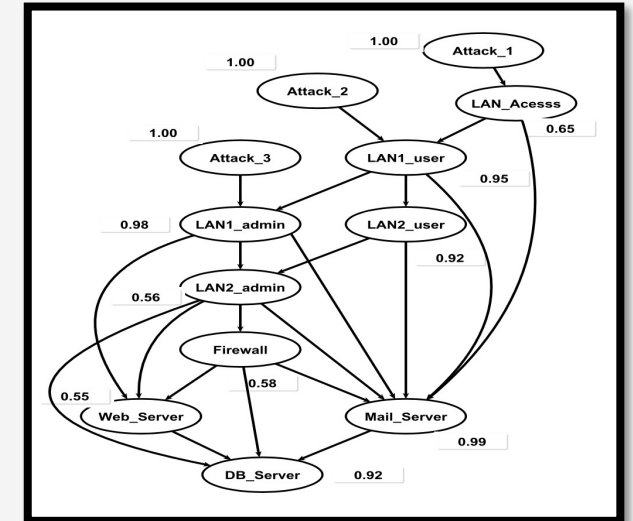
Stochastic Approximations

- Can yield different results in different runs
- Typically sampling-based
 - E.g., depending on different seeds for random numbers
- Should converge towards true result

Focus of this lecture

Side Note: *Variational Inference (VI)*

- Idea: Replace an optimisation or inference problem that is too difficult to solve with a simpler one, which offers guarantees in the form of upper / lower bounds w.r.t. the true result
 - Example: Assume further independencies such that a simpler model with lower tree width emerges
 - Lower example has width 3 (above: 7)
 - Restricts the search space for the true result to the space, which satisfies these independencies
 - Allows a lower bound, e.g., when maximising entropy
- Depending on the used method, VI yields a deterministic or stochastic approximation
 - Example: In lower model
 - Compute query with VE → deterministic
 - Compute query with sampling → stochastic



Query Answering Using Stochastic Simulation

- **Monte Carlo** methods
 - Repeated random sampling to get a numerical result
- Basic idea
 1. Draw N samples from a sampling distribution S
 2. Compute an approximate distribution $\hat{P}$ from samples
 3. Show that $\hat{P}$ converges to true distribution P
- Approximation quality depends on the number of samples N
- Sampling methods in this lecture
 - Direct sampling (in graph)
 - **Forward sampling, rejection sampling:** Sample from local distributions, *reject* samples that contradict evidence
 - **Likelihood weighting, importance sampling:** Sample from local distributions according to evidence, weight samples with likelihood
 - Markov Chain Monte Carlo (MCMC):
 - Sampling in a stochastic process, whose stationary distribution corresponds to the true distribution P
 - Methods: **Gibbs** and **Metropolis-Hastings**

Basic Idea of Sampling from a Probability Distribution

- Given some probability distribution P_R over random variables R and sampling target $R' \in R$, i.e., asking for $P(R')$
 - *Assumption:* Access to P_R to generate samples
- **Sample:** an observation (value) for each $R \in R$
 - I.e., sample = compound event for R
 - Sometimes called particle
- *Procedure*
 1. Generate a set of samples
 - For each sample: Generate a compound event for R based on P_R
 2. Based on the set of samples, estimate $P(R')$
 - Count how often does $R' = r'$ occur for each $r' \in \text{ran}(R')$ in the set
 - Normalise the counts over all $r' \in \text{ran}(R')$

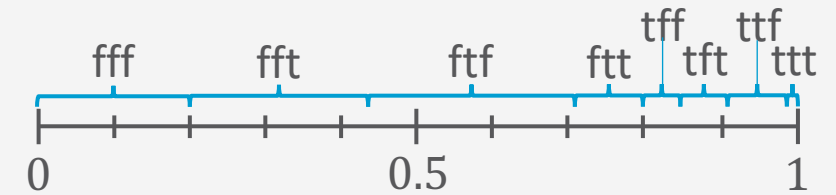
How do we do that? Can we use a random number generator?

<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
<i>false</i>	<i>false</i>	<i>false</i>	0.20
<i>false</i>	<i>false</i>	<i>true</i>	0.24
<i>false</i>	<i>true</i>	<i>false</i>	0.28
<i>false</i>	<i>true</i>	<i>true</i>	0.08
<i>true</i>	<i>false</i>	<i>false</i>	0.05
<i>true</i>	<i>false</i>	<i>true</i>	0.06
<i>true</i>	<i>true</i>	<i>false</i>	0.07
<i>true</i>	<i>true</i>	<i>true</i>	0.02

Sampling: 1. Generate Samples

a. Partition interval $[0,1]$ according to P_R

- Accumulating probabilities
- Formally, with $\text{ran}(\mathbf{R}) = \{\mathbf{r}_1, \dots, \mathbf{r}_m\}$ and a random but fixed order $(\mathbf{r}_1, \dots, \mathbf{r}_m)$:
 - $[0, x_1]$
 - $x_1 = P(\mathbf{r}_1)$
 - $(x_{i-1}, x_i], i \in \{2, \dots, m\}$
 - $x_i = x_{i-1} + P(\mathbf{r}_i)$
 - Last case ($i = m$): $(x_{m-1}, x_m] = (1 - P(\mathbf{r}_m), 1]$
- Example
 - Order $(\text{Epid}, \text{Travel}, \text{Sick}), \mathbf{R}' = \{\text{Travel}\}$
 - $[0, 0.20], (0.20, 0.20 + 0.24], (0.44, 0.44 + 0.28], \dots$



<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
false	false	false	0.20
false	false	true	0.24
false	true	false	0.28
false	true	true	0.08
true	false	false	0.05
true	false	true	0.06
true	true	false	0.07
true	true	true	0.02

Sampling: 1. Generate Samples

b. For a sample, generate a random number $v \in [0,1]$ and take the compound event that belongs to the partition, in which v is

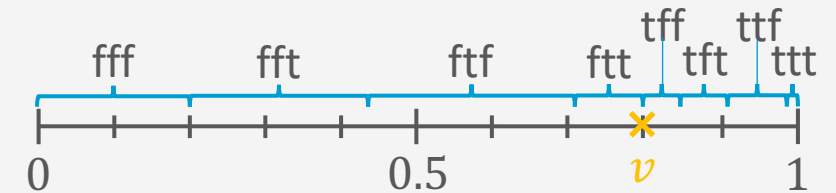
- Formally:

$$f(v) = \begin{cases} \mathbf{r}_1 & v \in [0, x_1], x_1 = P(\mathbf{r}_1) \\ \mathbf{r}_2 & v \in (x_1, x_2], x_2 = x_1 + P(\mathbf{r}_2) \\ \vdots & \vdots \\ \mathbf{r}_m & v \in (x_{m-1}, 1] \end{cases}$$

$$= \begin{cases} \mathbf{r}_1 & v \in [0, x_1], x_1 = P(\mathbf{r}_1) \\ \mathbf{r}_i & v \in (x_{i-1}, x_i], i \in \{2, \dots, m\}, x_i = x_{i-1} + P(\mathbf{r}_i) \end{cases}$$

- Example:

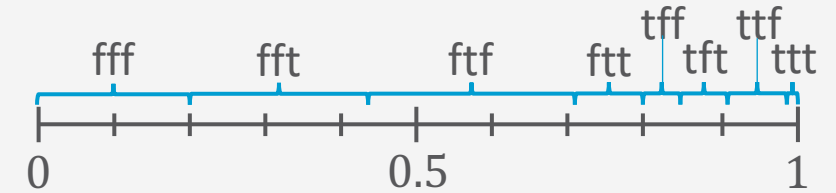
- Order $(Epid, Travel, Sick)$, $\mathbf{R}' = \{Travel\}$
- $v = 0.8 \rightarrow (\text{false}, \text{true}, \text{true})$



<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
false	false	false	0.20
false	false	true	0.24
false	true	false	0.28
false	true	true	0.08
true	false	false	0.05
true	false	true	0.06
true	true	false	0.07
true	true	true	0.02

Sampling: 2. Estimate $P(R')$ (Count Occurences)

2. Given a set of generated samples $\{r_k\}_{k=1}^N$
- Count how often the different values of R' occur over all r_k :
 - For each $r' \in \text{ran}(R')$: $n_{r'} = \sum_{k=1}^N 1 \mid \pi_{R'}(r_k) = r'$
 - Output: $\hat{P}(R') = \left(\frac{n_1}{n}, \dots, \frac{n_l}{n}\right)$
 - $n = \sum_{r' \in \text{ran}(R')} n_{r'}, l = |\text{ran}(R')|$
 - $\hat{P}(R') \approx P(R')$
 - Example:
 - Order $(Epid, Travel, Sick)$, $R' = \{Travel\}$
 - $(f, f, t), (t, t, f), (f, t, t), (t, f, t), \dots$
 - Assume $n_0 = 363, n_1 = 324, n = 687$



<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
false	false	false	0.20
false	false	true	0.24
false	true	false	0.28
false	true	true	0.08
true	false	false	0.05
true	false	true	0.06
true	true	false	0.07
true	true	true	0.02

<i>Travel</i>	$\hat{P}(Travel)$
false	$\frac{363}{687} = 0.528$
true	$\frac{324}{687} = 0.472$

$P(Travel)$
0.55
0.45

Sampling: Generalisation

- Generalisation: Estimate expectation of some function $f(\mathbf{R})$ relative to a distribution $P(\mathbf{R})$

$$E_{P(\mathbf{R})}[f(\mathbf{R})] = \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} f(\mathbf{r})P(\mathbf{r})$$

- Generate a set of N samples, estimating value of f or its expectation
- Aggregate the results

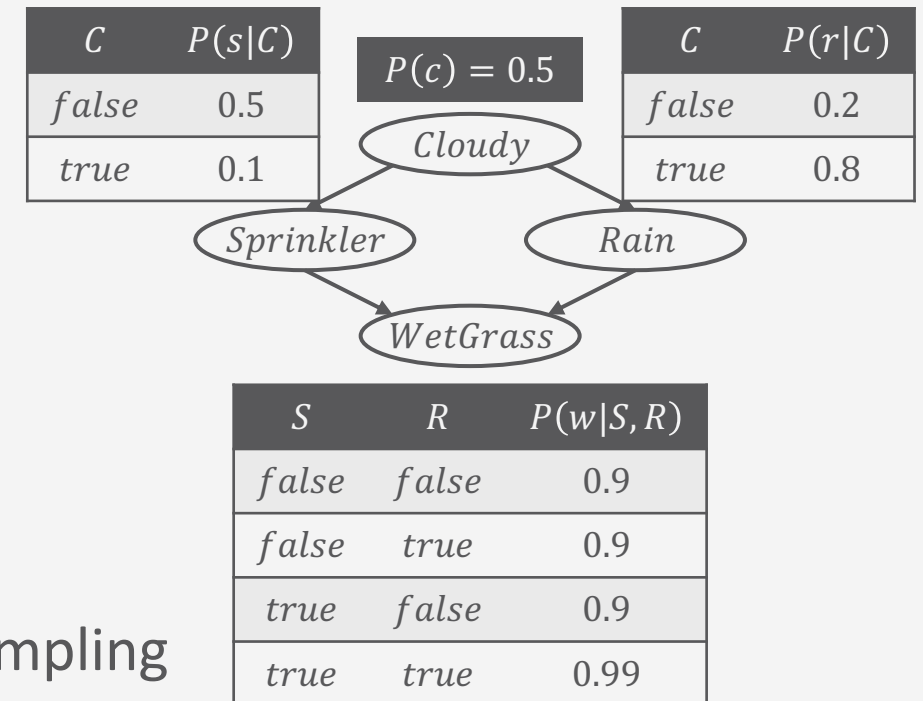
$$E_P[f(\mathbf{r})] \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{r}_i)$$

- Can choose f to be indicator function $\mathbf{1}$ that is 1 if $\mathbf{R} = \mathbf{r}$ and 0 otherwise (which is what happens on the following slides)
- Accuracy usually depends on number of samples N
 - Because then the *law of large numbers* applies

Direct Sampling

Rejection Sampling, Likelihood Weighting, Importance Sampling

Lifted Importance Sampling



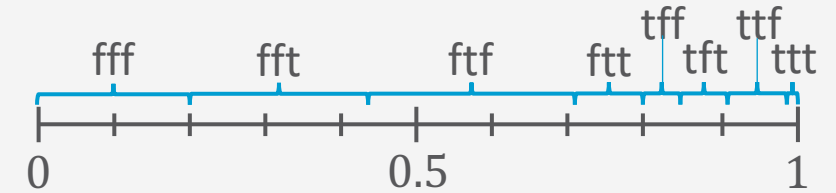
Direct Sampling: Forward & Rejection Sampling

- Given a full joint P_R over random variables R ,
evidence e , possibly empty, and query terms $R' \in R$
 - Find approximate answer to query $P(R'|e)$ using sampling

Procedure

1. Generate a set of samples over R

- Independent of e
- If only P_R given: Partition $[0,1]$ based on P_R , generate samples
- If factorised model given: Use factorisation during sampling



How do we
do that?

2. Given the set of samples, estimate $P(R'|e)$

- Count how often $R' = r'$ occurs for each $r' \in \text{ran}(R')$ in the set that are consistent with e
 - Reject (drop) those samples that do not agree with e

How does
this work?

If $e = \emptyset$, called *forward sampling*
If $e \neq \emptyset$, called *rejection sampling*

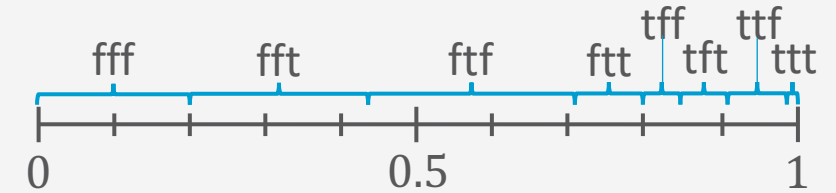
Epid	Travel	Sick	P
false	false	false	0.20
false	false	true	0.24
false	true	false	0.28
false	true	true	0.08
true	false	false	0.05
true	false	true	0.06
true	true	false	0.07
true	true	true	0.02

Rejecting Samples

- Given a set of samples $\{\mathbf{r}_k\}_{k=1}^N$
 - Count how often $\mathbf{R}' = \mathbf{r}'$ occurs for each $\mathbf{r}' \in \text{ran}(\mathbf{R}')$ in the set that are consistent with $\mathbf{e}$
 - Formally, only consider $\{\mathbf{r}_k\}_{k=1}^{N'} = \{\mathbf{r}_k \mid \pi_{\mathbf{rv}(\mathbf{e})}(\mathbf{r}_k) = \mathbf{e}, k \in \{1, \dots, N\}\}$
- Example: $P(\text{Travel} \mid \text{sick})$
 - Order $(\text{Epid}, \text{Travel}, \text{Sick})$, $\mathbf{e} = \{\text{sick}\}$, $\mathbf{R}' = \{\text{Travel}\}$
 - $(f, f, t), (f, t, f), (f, t, t), (t, f, t), \dots$
 - Assume $n_0 = 201, n_1 = 74, n = 275$

Reject those samples that are not consistent with $\mathbf{e}$

Next part: How can we use a factorised model for this?



<i>Epid</i>	<i>Travel</i>	<i>Sick</i>	<i>P</i>
false	false	false	0.20
false	false	true	0.24
false	true	false	0.28
false	true	true	0.08
true	false	false	0.05
true	false	true	0.06
true	true	false	0.07
true	true	true	0.02

<i>Travel</i>	$\hat{P}(\text{Travel} \text{sick})$	$P(\text{Travel} \text{sick})$
false	$\frac{201}{275} = 0.731$	0.75
true	$\frac{74}{275} = 0.269$	0.25

Remember: Bayes Nets (BNs)

- **Bayesian network** B : Directed, acyclic graph $B = (V, E)$
 - Each $V \in V$ labelled with a *conditional probability table* (CPT) $P(V|Pa(V))$
 - Semantics
 - Each $V \in V$ corresponds to a random variable $R \in \mathbf{R}$
 - B represents the full joint probability distribution

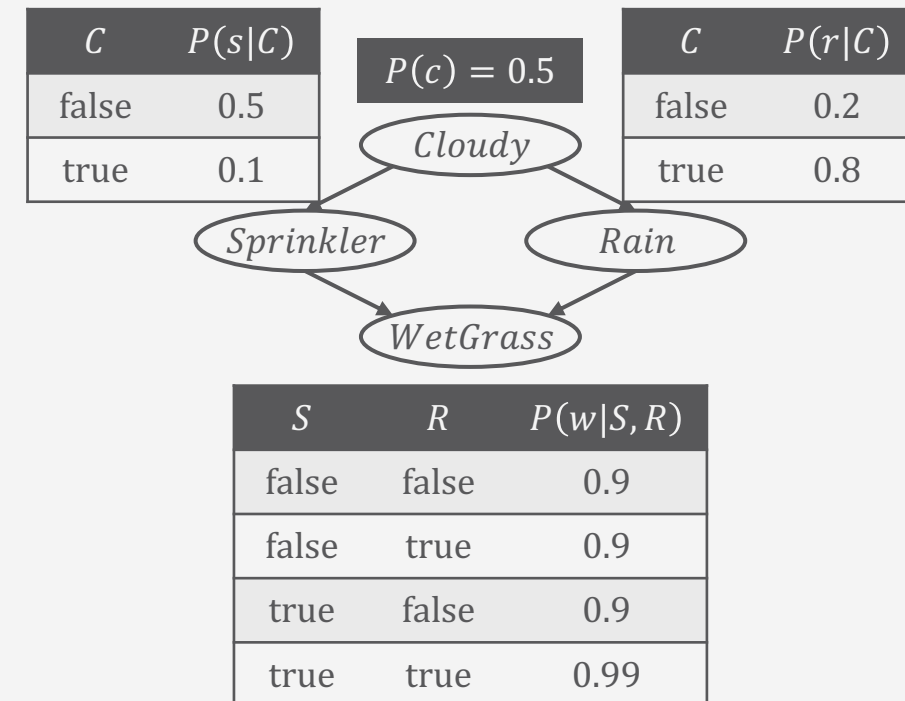
$$P_B = P(\mathbf{R}) = \prod_{R \in \mathbf{R}} P(R|Pa(R))$$

Why BNs?

- Given a probability distribution, we can sample from it
- In CPTs: Parent values determine which probability distribution holds
- If sampling from roots to leaves (along a *topological order*), we always have parent values for CPTs (enabled by acyclicity)

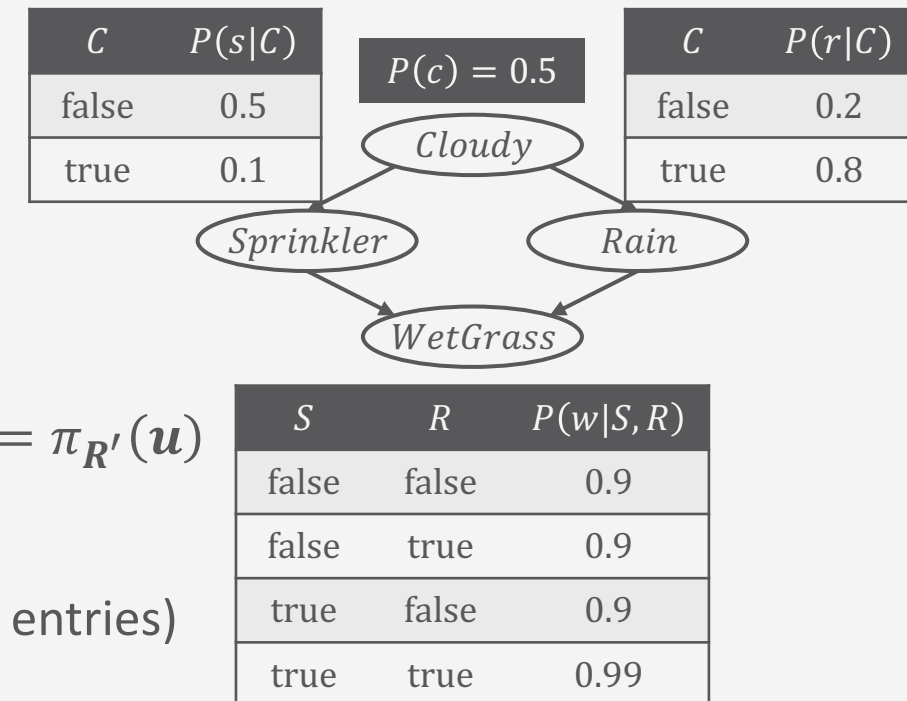
BN represents full joint $P_{C,S,R,W}$

$$\begin{aligned} P_{C,S,R,W} &= P(C, S, R, W) \\ &= P(C)P(S|C)P(R|C)P(W|S, R) \end{aligned}$$



Direct Sampling: Forward + Rejection Sampling in BNs

- Given a BN B over R , a **topological order** $\mathcal{U} = (U_1, \dots, U_n)$ over R , **evidence** e , possibly empty, and query terms R'
 - Build up a full sample along the graph, sampling from CPTs
- Procedure: Do N times
 - Generate sample $\mathbf{u}$ along $\mathcal{U}$, i.e.,
 - For each $U_i \in \mathcal{U}$, sample a value u_i given previously sampled values for its parent nodes $\text{Pa}(U_i)$
 - In a *count vector* with $|\text{ran}(R)|$ entries, initially 0:
 - If $\mathbf{u}$ consistent with e ($\pi_{\text{rv}(e)}(\mathbf{u}) = e$), increment counter at $\mathbf{r}' = \pi_{R'}(\mathbf{u})$
 - Output: $\hat{P}(R) = \left(\frac{n_1}{n}, \dots, \frac{n_l}{n}\right)$
 - Count vector with its entries normalised (divide each by sum of entries)



1. Generate Sample

- Given a topological order $\mathcal{U} = (U_1, \dots, U_n)$
- For each U_i , sample a value from its CPT $P(U_i \mid \text{Pa}(U_i))$ given the sampled values of its parent nodes $\text{Pa}(U_i)$, i.e., sample from

$$P(U_i \mid \pi_{\text{Pa}(U_i)}(u_1, \dots, u_{i-1}))$$

- Is a probability distribution, i.e., partition $[0,1]$ according to it
- Example: $\mathcal{U} = (C, S, R, W)$
 - Sample from $P(C) = (0.5, 0.5) \rightarrow \text{true}$
 - Sample from $P(S \mid c) = (0.1, 0.9) \rightarrow \text{true}$
 - Sample from $P(R \mid c) = (0.8, 0.2) \rightarrow \text{true}$
 - Sample from $P(W \mid s, r) = (0.99, 0.01) \rightarrow \text{true}$

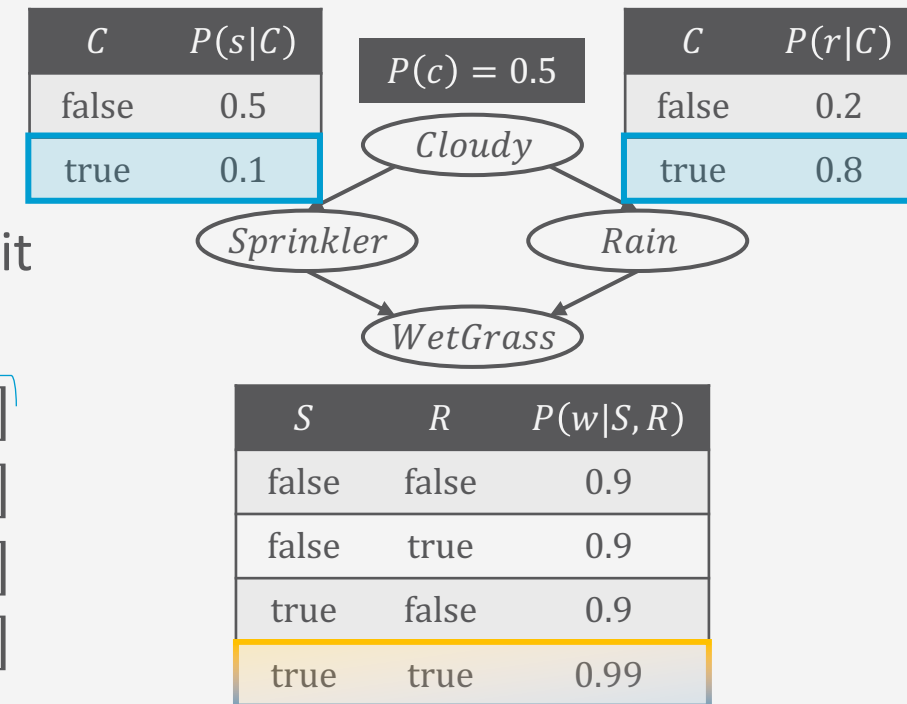
Sample

$\rightarrow [c, \text{ }, \text{ }, \text{ }]$

$\rightarrow [c, s, \text{ }, \text{ }]$

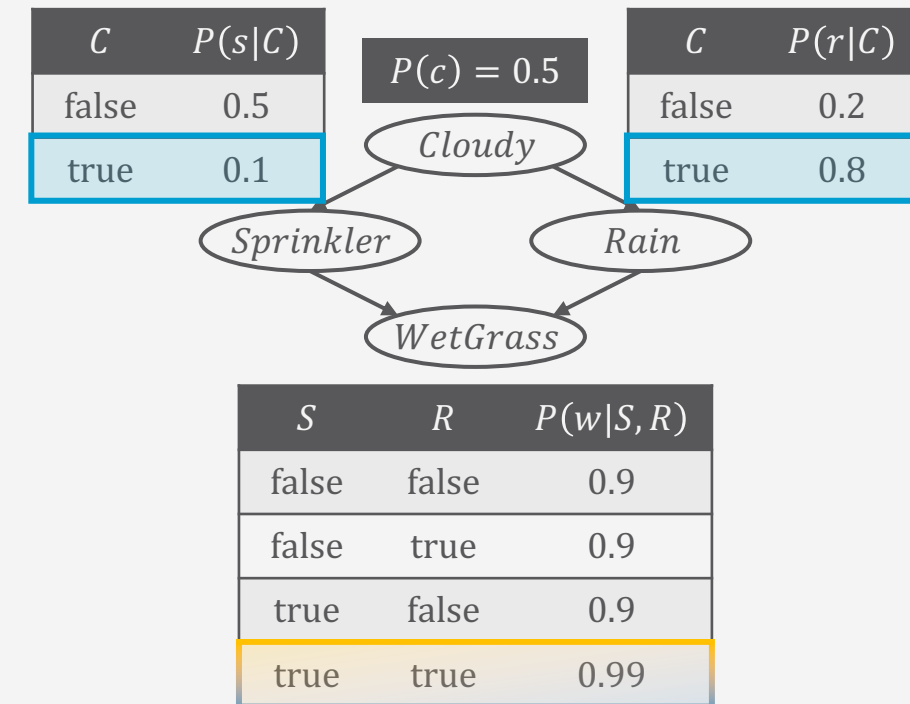
$\rightarrow [c, s, r, \text{ }]$

$\rightarrow [c, s, r, w]$



2. Count, and Output

- Given $\mathbf{u}$ and query terms R'
- In a count vector N with $|\text{ran}(R)|$ entries, increment the counter at $r' = \pi_{R'}(\mathbf{u})$ (if $\mathbf{u}$ is consistent with e)
- After having sampled N times, normalise entries in N by dividing each entries by the sum of the entries in N
- Example: Query term *Rain*
 - Sample is $[c, s, r, w]$
 - Increment counter for r
 - Assume evidence $WetGrass = false$, i.e., $\neg w$
 - Reject (drop) sample, as $w \neq \neg w$
 - If $N = \begin{matrix} \text{true} & \text{false} \\ [321, 310] \end{matrix}$, output is $N = [0.509, 0.491]$



Rejection Sampling

RejectionSampling($R', e, B, N, \mathcal{U}$)

Vector N of length $|\text{ran}(R)|$, initially 0

for $t = 1, \dots, N$ **do**

 Sample $\mathbf{u} \leftarrow \text{PriorSample}(B, \mathcal{U})$

if $\pi_{\text{rv}(e)}(\mathbf{u}) = e$ **then**

$N[\mathbf{r}'] \leftarrow N[\mathbf{r}'] + 1$ with $\mathbf{r}' = \pi_{R'}(\mathbf{u})$

return Normalise(N)

▸ Forward sampling if $e = \emptyset$

▸ Stores counters for each $\mathbf{r}' \in \text{ran}(R')$

▸ Always true with $e = \emptyset$

▸ Divide each entry by sum of entries

PriorSample($B, \mathcal{U}$)

Empty Sample $\mathbf{u} \leftarrow (\perp_1, \dots, \perp_{\text{len}(\mathcal{U})})$

for $i = 1, \dots, \text{len}(\mathcal{U})$ **do**

$\mathbf{u}[i] \leftarrow \text{Sample value from } P(U_i | \pi_{\text{Pa}(U_i)}(\mathbf{u}[1:i-1]))$

return $\mathbf{u}$

Rejection Sampling

Forward Sampling: Analysis

- Probability that **PriorSample** generates a particular event:

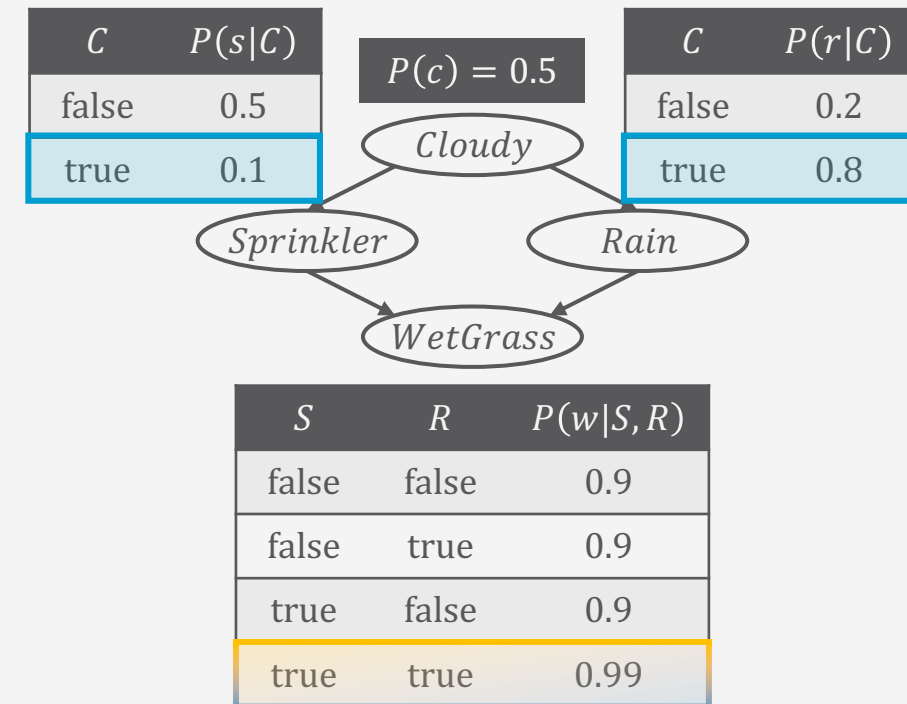
$$S_{PS}(u_1, \dots, u_n) = \prod_{i=1}^n P(u_i \mid \text{Pa}(U_i)) = P(u_1, \dots, u_n)$$

- I.e., true probability
- Then, it holds

$$\begin{aligned} \lim_{N \rightarrow \infty} \hat{P}(u_1, \dots, u_n) &= \lim_{N \rightarrow \infty} \frac{N_{PS}(u_1, \dots, u_n)}{N} \\ &= S_{PS}(u_1, \dots, u_n) \\ &= P(u_1, \dots, u_n) \end{aligned}$$

- $N_{PS}(u_1, \dots, u_n)$ number of samples generated for $u_1, \dots, u_n$
- That is, estimates derived from **PriorSample** are **consistent**
- I.e., $\hat{P}(u_1, \dots, u_n) \approx P(u_1, \dots, u_n)$

$$\begin{aligned} S_{PS}(c, s, r, w) \\ &= 0.5 \cdot 0.1 \cdot 0.8 \cdot 0.99 \\ &= 0.0396 \end{aligned}$$

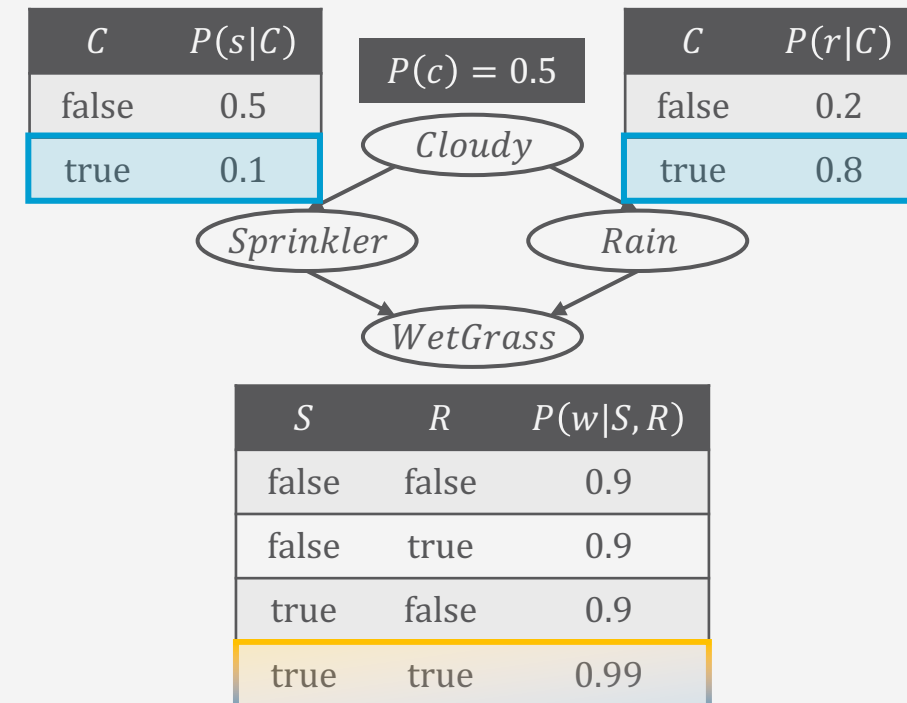


Rejection Sampling: Analysis

- With $N_{PS}(\mathbf{r}', \mathbf{e})$ the number of samples that agree with $\mathbf{r}', \mathbf{e}$, it holds:

$$\begin{aligned}\hat{P}(\mathbf{r}' | \mathbf{e}) &= \frac{1}{Z} N_{PS}(\mathbf{r}', \mathbf{e}) \\ &= \frac{N_{PS}(\mathbf{r}', \mathbf{e})}{N_{PS}(\mathbf{e})} \\ &\approx \frac{P(\mathbf{r}', \mathbf{e})}{P(\mathbf{e})} \\ &= P(\mathbf{r}' | \mathbf{e})\end{aligned}$$

- Estimation using RejectionSampling consistent



Brief Look into PAC Estimators

- Quality of result depends on number of samples N
- **PAC**: With probability $1 - \delta$, the error is bounded by ε
- Show estimate $\hat{P}(\mathbf{u})$ close to truth $P(\mathbf{u})$ using so-called Hoeffding bound
 - *Hoeffding bound*: Given a sequence of Bernoulli trials $\{r[1], \dots, r[N]\}$, $\hat{P} = \frac{1}{N} \sum_i r[i]$, and success probability p :

$$P(\hat{P} > p + \varepsilon) \leq e^{-2N\varepsilon^2}$$

$$P(\hat{P} < p - \varepsilon) \leq e^{-2N\varepsilon^2}$$

- If samples are independent Bernoulli trials, each with probability $P(\mathbf{u})$, then

$$P(\hat{P}(\mathbf{u}) \notin [P(\mathbf{u}) - \varepsilon, P(\mathbf{u}) + \varepsilon]) \leq 2e^{-2N\varepsilon^2}$$

- Setting $2e^{-2N\varepsilon^2} \leq \delta$, to estimate a query $P(\mathbf{e})$ with (ε, δ) reliability: $N \geq \frac{\ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2}$

Brief Look into PAC Estimators

- Quality of result depends on number of samples N
- **PAC**: With probability $1 - \delta$, the error is bounded by ε
- Show estimate $\hat{P}(\mathbf{u})$ has relative error ε of true value $P(\mathbf{u})$ using so-called Chernoff bound

What is a problem with sampling like this?

- *Chernoff bound*: Given a sequence of Bernoulli trials $\{r[1], \dots, r[N]\}$, $\hat{P} = \frac{1}{N} \sum_i r[i]$, and success probability p :

$$P(\hat{P} > p(1 + \varepsilon)) \leq e^{-Np\varepsilon^2/3}$$

$$P(\hat{P} < p(1 - \varepsilon)) \leq e^{-Np\varepsilon^2/2}$$

- If samples are independent Bernoulli trials, each with probability $P(\mathbf{u})$, then

$$P(\hat{P}(\mathbf{u}) \notin [P(\mathbf{u})(1 - \varepsilon), P(\mathbf{u})(1 + \varepsilon)]) \leq 2e^{-NP(\mathbf{u})\varepsilon^2/3}$$

- Setting $2e^{-NP(\mathbf{u})\varepsilon^2/3} \leq \delta$, to guarantee error probability δ given error bound ε : $N \geq 3 \frac{\ln\left(\frac{2}{\delta}\right)}{P(\mathbf{u})\varepsilon^2}$

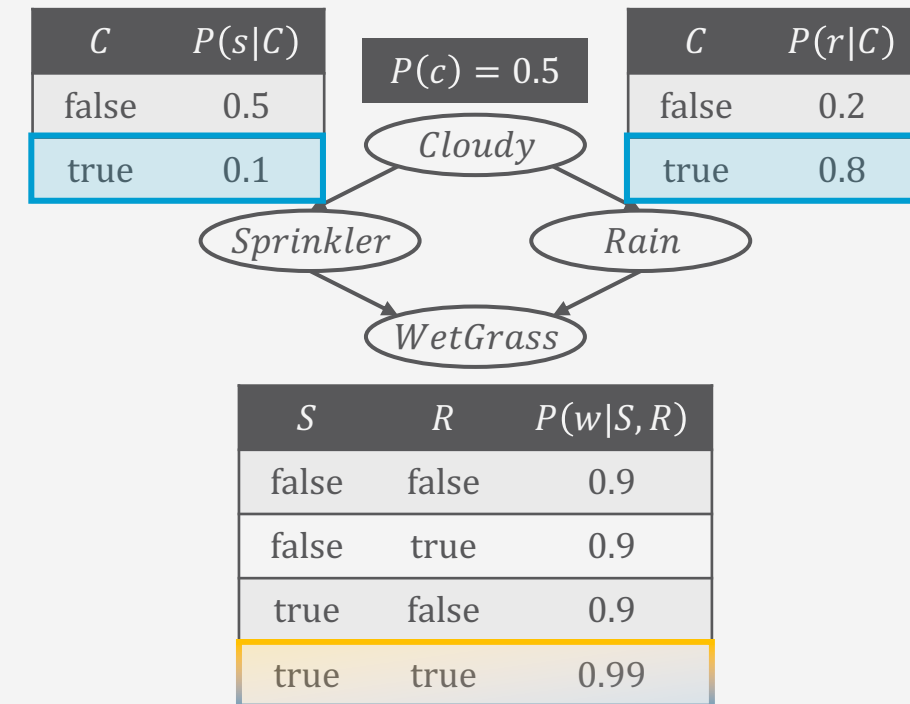
Rejection Sampling: Problem

- Example
 - Estimate $P(\text{Rain} \mid \text{Sprinkler} = \text{true})$ with 100 Samples:
 - 73 samples: $\text{Sprinkler} = \text{false} \rightarrow \text{reject}$
 - 27 samples: $\text{Sprinkler} = \text{true} \rightarrow \text{count}$
 - 8 samples: $\text{Rain} = \text{true}$
 - 19 samples: $\text{Rain} = \text{false}$
 - Output:

$$P(\text{Rain} \mid \text{Sprinkler} = \text{true})$$

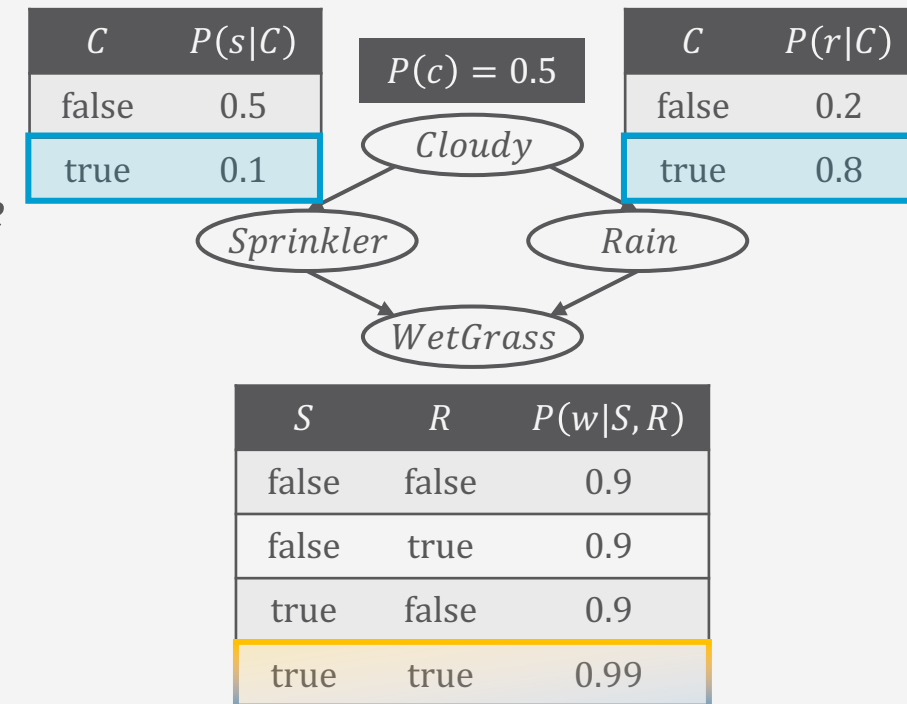
$$= \text{Normalise}((8,19))$$

$$= (0.296, 0.704)$$
 - Too many samples rejected
 - Too many samples needed to get a reliable result



Rejection Sampling: Problem

- Sampling hopelessly expensive if $P(e)$ small
 - Because probability of generating samples consistent with e very small
 - $P(e) = 0.001, N = 10000 \rightarrow$ around 10 samples not rejected
 - E.g., holds for most symptoms in medical diagnosis systems
 - $P(e)$ gets exponentially smaller with more observations in e
- Let N^* be the number of samples not rejected that we want to have
- Then, we need to generate $N = \frac{N^*}{P(e)}$ samples
 - $N^* = 10000, P(e) = 0.001 \rightarrow N = 10,000,000$



Likelihood Weighting

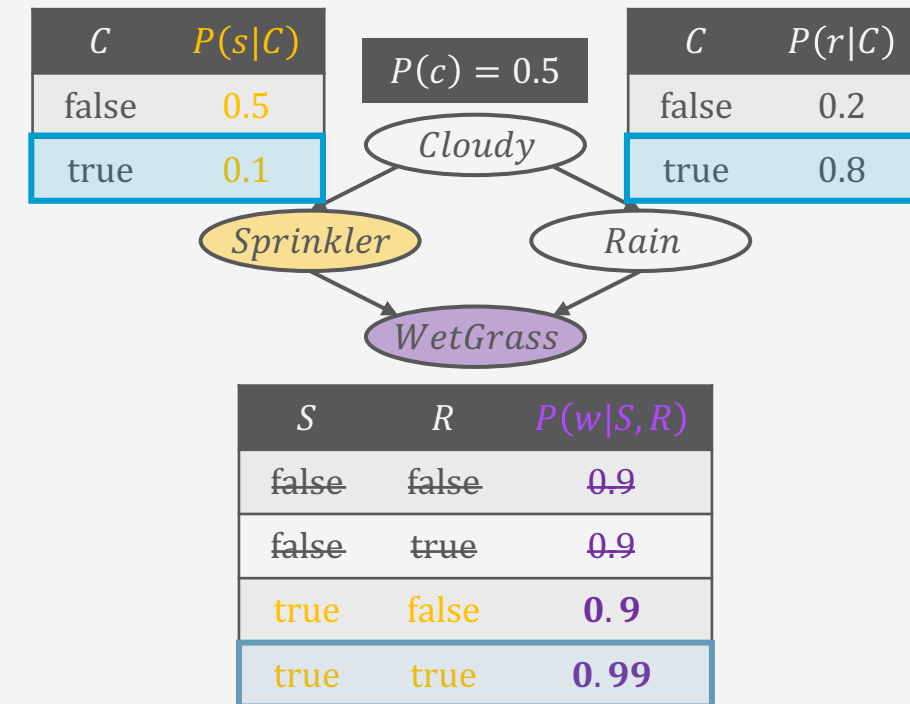
- Goal
 - Avoid inefficiency of rejection sampling
- Idea
 - Generate only events consistent with evidence e
 - Each event is weighted by likelihood that the event accords to the evidence

Likelihood Weighting: Example

- $P(R|S = \text{true}, W = \text{true})$?
 - Topological order: (C, S, R, W)
- Sampling (repeat N times)
 - Weight w of sample is set to 1.0
 - Sample C from $P(C) = (0.5, 0.5) \rightarrow \text{true}$
 - S is an evidence variable with value **true**

$$w \leftarrow w \cdot P(S = \text{true} | C = \text{true}) = 0.1$$
 - Sample R from $P(R | C = \text{true}) = (0.8, 0.2) \rightarrow \text{true}$
 - W is an evidence variable with value **true**

$$w \leftarrow w \cdot P(W = \text{true} | S = \text{true}, R = \text{true}) = 0.099$$
 - **[true, true, true, true]** with weight 0.099
 - For estimating, accumulate weights for $R = \text{true}$ and $R = \text{false}$
 - Above sample goes toward $R = \text{true}$ with weight 0.099
 - Normalise (= divide by sum of weights)

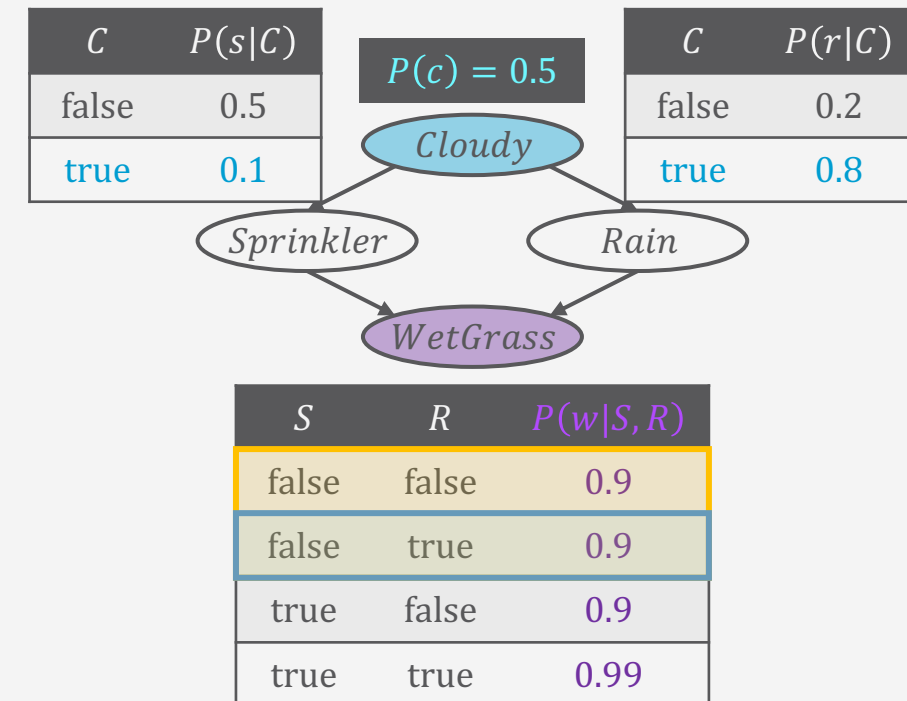


Likelihood Weighting: Example

- $P(R|C = \text{true}, W = \text{true})$?
 - Topological order: (C, S, R, W)
- Sampling (repeat N times)
 - Weight w of sample is set to 1.0
 - C is an evidence variable with value **true**

$$w \leftarrow w \cdot P(C = \text{true}) = 1.0 \cdot 0.5 = 0.5$$
 - Sample S from $P(S|C = \text{true}) = (0.1, 0.9) \rightarrow$ **false**
 - Sample R from $P(R|C = \text{true}) = (0.8, 0.2) \rightarrow$ **true**
 - W is an evidence variable with value **true**

$$w \leftarrow w \cdot P(W = \text{true}|S = \text{false}, R = \text{true}) = 0.5 \cdot 0.9 = 0.45$$
 - [**true**, **false**, **true**, **true**] with weight 0.45
 - For estimating, accumulate weights for $R = \text{true}$ and $R = \text{false}$
 - Above sample goes toward $R = \text{true}$ with weight 0.45
 - Normalise (= divide by sum of weights)



Likelihood Weighting: Algorithm

function LikelihoodWeighting(R', e, B, N) **returns** an estimate of $P(R'|e)$

local variables:

W , a vector of weighted counts over the range values of R' , initially 0

for $j = 1$ **to** N **do**

$r, w \leftarrow \text{WeightedSample}(B, e)$

$W[r'] \leftarrow W[r'] + w$ where r' are the values of R' in r

return Normalise(W)

‣ divide each entry by sum of all entries

function WeightedSample(B, e) **returns** a compound event and a weight

$r \leftarrow$ an event with $n = |rv(B)|$ elements; $w \leftarrow 1$

for $i = 1$ **to** n **do**

‣ follows topological order

if R_i has a value r_i in e **then**

$w \leftarrow w \cdot P(r_i | \text{pa}(R_i))$; set r_i in r

else

$r_i \leftarrow$ a random sample from $P(R_i | \text{pa}(R_i))$

return r, w

Likelihood Weighting

Likelihood Analysis

What if we cannot
sample from P ?

- Sampling probability for WeightedSample
 - i goes through $\mathbf{r} \setminus \mathbf{e}$
 - Takes only evidence in ancestors into consideration
- Weight for a given sample $\mathbf{r}, \mathbf{e}$
 - j goes through $\mathbf{e}$
- Weighted sampling probability is

$$S_{WS}(\mathbf{r}, \mathbf{e}) = \prod_{i=1}^l P(r_i | \text{Pa}(R_i))$$

$$w(\mathbf{r}, \mathbf{e}) = \prod_{j=1}^k P(e_j | \text{Pa}(E_j))$$

$$S_{WS}(\mathbf{r}, \mathbf{e}) \cdot w(\mathbf{r}, \mathbf{e}) = \prod_{i=1}^l P(r_i | \text{pa}(R_i)) \cdot \prod_{j=1}^k P(e_j | \text{pa}(E_j)) = P(\mathbf{r}, \mathbf{e})$$

- Last step by semantics of BN
- Hence, likelihood weighting returns consistent estimates

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

Importance Sampling

- Remember: Estimates expectation of a function $f(\mathbf{r})$ relative to some distribution $P(\mathbf{R})$

$$E_{P(\mathbf{R})}[f(\mathbf{R})] = \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{R})} f(\mathbf{r})P(\mathbf{r})$$

- $P(\mathbf{R})$ typically called **target distribution**
- Estimate this expectation by generating samples $\mathbf{r}_i$ from P and then estimating

$$E_P[f(\mathbf{r})] \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{r}_i)$$

- What we have done so far*
- If generating samples from P is **hard**, use a (simpler) **proposal distribution** Q instead

Using a Proposal Distribution

- Condition: Proposal distribution Q dominates P
 - I.e., $Q(\mathbf{r}) > 0$ whenever $P(\mathbf{r}) > 0$
 - Q may not ignore any states that have non-zero probability with P
 - Specifically, support of Q has to include support of P
 - Support for a distribution S are all points $\mathbf{r}$ s. t. $S(\mathbf{r}) > 0$
 - In general, Q can be arbitrary but computational performance highly depends on how similar Q to P is
 - E.g., want probabilities close to zero in $Q(\mathbf{r})$ only if $f(\mathbf{r})P(\mathbf{r})$ also very small
- Generate samples from Q instead of P
 - Cannot average f -values of samples generated
→ Adjust estimator to compensate for incorrect sampling distribution

$$E_{P(\mathbf{R})}[f(\mathbf{R})] = \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{R})} f(\mathbf{r})P(\mathbf{r})$$

Unnormalised Importance Sampling

- How to adjust:

$$\begin{aligned}
 E_{P(R)}[f(R)] &= \sum_{r \in \text{ran}(R)} f(r)P(r) \\
 &= \sum_{r \in \text{ran}(R)} Q(r)f(r) \frac{P(r)}{Q(r)} \\
 &= E_{Q(R)} \left[f(R) \frac{P(R)}{Q(R)} \right]
 \end{aligned}$$

- Adjustment: $\frac{P(R)}{Q(R)}$

Assumes that P is known

- Generate a set of samples from Q and then estimate

$$E_P[f(r)] \approx \frac{1}{N} \sum_{i=1}^N f(r_i) \frac{P(r_i)}{Q(r_i)} = \frac{1}{N} \sum_{i=1}^N f(r_i) w(r_i)$$

$$w(r_i) \stackrel{\text{def}}{=} \frac{P(r_i)}{Q(r_i)}$$

When P is known: Example

- Interpret likelihood weighting as importance sampling
 - Model without evidence set is P , model with evidence is Q
 - Samples are weighted according to probabilities in CPTs
 - Probability of a sample $\mathbf{v}_i$ in model B conformant with evidence $\mathbf{e}$, i.e., P :

$$P(\mathbf{v}_i|\mathbf{e}) = \frac{P(\mathbf{v}_i, \mathbf{e})}{P(\mathbf{e})} = \frac{\prod_{r_j \in (\mathbf{v}_i \cup \mathbf{e})} P(r_j|\text{pa}(R_j))}{P(\mathbf{e})}$$

- Probability of a sample $\mathbf{v}_i$ sampled in model B with evidence set, i.e., Q :

$$Q(\mathbf{v}_i) = \prod_{r_j \in \mathbf{v}_i} P(r_j|\text{pa}(R_j))$$

- Weight

$$w(\mathbf{v}_i) = \frac{P(\mathbf{v}_i|\mathbf{e})}{Q(\mathbf{v}_i)} = \frac{\prod_{r_j \in (\mathbf{v}_i \cup \mathbf{e})} P(r_j|\text{pa}(R_j))}{P(\mathbf{e}) \prod_{r_j \in \mathbf{v}_i} P(r_j|\text{pa}(R_j))} = \frac{\prod_{r_j \in \mathbf{e}} P(r_j|\text{pa}(R_j))}{P(\mathbf{e})}$$

- $P(\mathbf{e})$ identical for all samples $\rightarrow$ okay to ignore, i.e.,

$$w(\mathbf{v}_i) = \prod_{r_j \in \mathbf{e}} P(r_j|\text{pa}(R_j))$$

When P is not known

- Most common reason for sampling from Q :
 P only known up to normalising constant Z
 - Access to a function $\tilde{P}$ that is not a normalised distribution but $\tilde{P}(\mathbf{R}) = P(\mathbf{R}) \cdot Z$

$$w(\mathbf{r}_i) \stackrel{\text{def}}{=} \frac{\cancel{P(\mathbf{r}_i)}}{Q(\mathbf{r}_i)}$$

- Normalised Importance Sampling

- Define $w(\mathbf{r}_i)$ using $\tilde{P}$: $w(\mathbf{r}_i) \stackrel{\text{def}}{=} \frac{\tilde{P}(\mathbf{r}_i)}{Q(\mathbf{r}_i)}$
- Estimation no longer works (no probability distribution)

$$\cancel{E_P[f(\mathbf{r})] \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{r}_i) \frac{P(\mathbf{r}_i)}{Q(\mathbf{r}_i)}}$$

$$\begin{aligned} E_{P(\mathbf{R})}[f(\mathbf{R})] &\neq \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} f(\mathbf{r}) \tilde{P}(\mathbf{r}) \\ &= \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} Q(\mathbf{r}) f(\mathbf{r}) \frac{\tilde{P}(\mathbf{r})}{Q(\mathbf{r})} \\ &\neq E_{Q(\mathbf{R})} \left[f(\mathbf{R}) \frac{\tilde{P}(\mathbf{r})}{Q(\mathbf{R})} \right] \end{aligned}$$

Normalised Importance Sampling

- Trick: Consider $w(\mathbf{R})$ as a random variable with expected value Z

$$E_{Q(\mathbf{R})}[w(\mathbf{R})] = \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} Q(\mathbf{r})w(\mathbf{r}) = \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} Q(\mathbf{r}) \frac{\tilde{P}(\mathbf{r})}{Q(\mathbf{r})} = \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} \tilde{P}(\mathbf{r}) = Z$$

- Given $E_{Q(\mathbf{R})}[w(\mathbf{R})] = Z$

$$\begin{aligned} E_{P(\mathbf{R})}[f(\mathbf{R})] &= \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} f(\mathbf{r})P(\mathbf{r}) = \frac{1}{Z} \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} Q(\mathbf{r})f(\mathbf{r}) \frac{\tilde{P}(\mathbf{r})}{Q(\mathbf{r})} \\ &= \frac{1}{Z} E_{Q(\mathbf{R})}[f(\mathbf{R})w(\mathbf{R})] = \frac{E_{Q(\mathbf{R})}[f(\mathbf{R})w(\mathbf{R})]}{E_{Q(\mathbf{R})}[w(\mathbf{R})]} \end{aligned}$$

- Use estimator for numerator and denominator

$$E_P[f(\mathbf{r})] \approx \frac{\sum_{i=1}^N f(\mathbf{r}_i)w(\mathbf{r}_i)}{\sum_{i=1}^N w(\mathbf{r}_i)}$$

Sampling in Undirected Models

- Assuming we have a proposal distribution $Q(\mathbf{R})$ for a factor-based model F , which allows for easy sampling, we can generate samples $\mathbf{r}_i$ and weight them accordingly:

$$w(\mathbf{r}_i) = \frac{\tilde{P}(\mathbf{r}_i)}{Q(\mathbf{r}_i)}$$

- $\tilde{P}(\mathbf{r}_i) = \prod_f \phi_f(\pi_{rv(f)}(\mathbf{r}_i))$, i.e., product of potentials that $\mathbf{r}_i$ maps to in each f
- If we do this for all samples, we estimate Z (or $P(\mathbf{e})$):

$$E_{P(\mathbf{R})}[w(\mathbf{R})] \approx \sum_{\mathbf{r} \in \text{ran}(\mathbf{R})} \frac{\tilde{P}(\mathbf{r}_i)}{Q(\mathbf{r}_i)} = Z$$

- If are interested in $P(\mathbf{r}'|\mathbf{e})$:

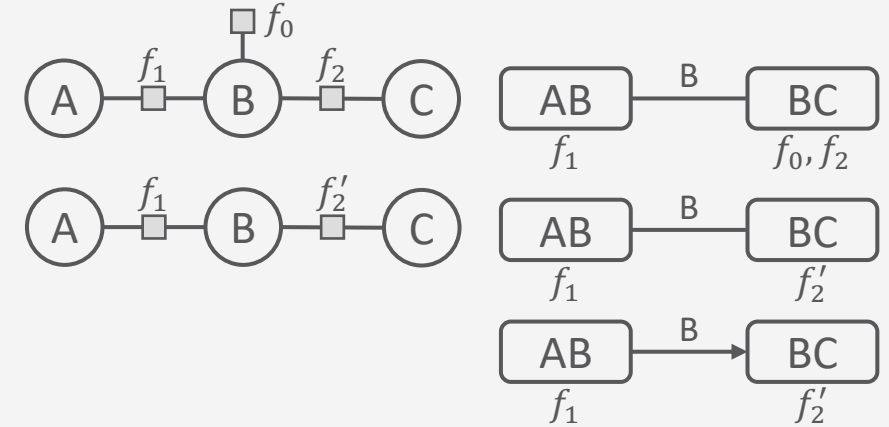
$$E_P[f(\mathbf{r}')] \approx \frac{\sum_{i=1}^N f(\mathbf{r}') \frac{\tilde{P}(\mathbf{r}')}{Q(\mathbf{r}')}}{\sum_{i=1}^N \frac{\tilde{P}(\mathbf{r}_i)}{Q(\mathbf{r}_i)}}$$

Where To Get Q From? – An Idea

- Given a factor-based model F
 1. Turn F into a model F' s. t. factors are over maximal cliques
 - In jtree: Multiply all factors in each local model G_i of cluster $\mathcal{C}_i$ into a single factor $f_i = \phi_i(\mathcal{C}_i)$
 2. Turn F' into a “sort-of” BN F^{BN} by
 - a. Choose one cluster as root and direct all edges away from that cluster $\rightarrow$ “directed jtree”
 - b. Normalise such that
 - i. Root cluster: marginal over all random variables (potentials sum to 1)
 - ii. All other clusters: conditional on separator random variables of “parent” in directed jtree
 - $\rightarrow$ Enforces a factorisation into (conditional) probability distributions with $Z = 1$
 - $\rightarrow$ Fixes a form of topological ordering over random variables in F
 - Root cluster random variables first, followed by random variables as they are visited in the directed jtree
 3. Sample from $Q = F^{BN}$, e.g., using likelihood weighting

Example

- Model F with jtree J
- Model F' , max. cliques: $f'_2 = f_2 \cdot f_0$
- Choose one as root, e.g., $\mathcal{C}_1 = \{AB\}$
- Normalise
 - $\mathcal{C}_1$ such that f_1 is a marginal distribution f_1^{BN}
 - $\mathcal{C}_2$ such that f'_2 is a distribution f_2^{BN} conditional on B

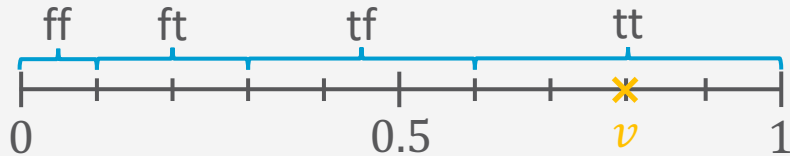


A	B	f_1	A	B	f_1^{BN}
false	false	1	false	false	0.1
false	true	2	false	true	0.2
true	false	3	true	false	0.3
true	true	4	true	true	0.4

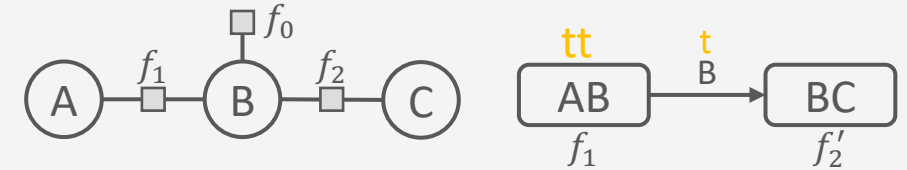
B	C	f'_2	B	C	f_2^{BN}
false	false	6	false	false	0.6
false	true	4	false	true	0.4
true	false	4	true	false	0.8
true	true	1	true	true	0.2

Example

- Ordering of AB, C
- Sample values for AB from f_1^{BN}
 - Since it is a distribution
 - Generate a random number v between 0 and 1 and take values for AB where v lies in the corresponding interval



- E.g., $v = 0.8 \rightarrow tt$
- Map to separator random variables
 $\rightarrow B = t$

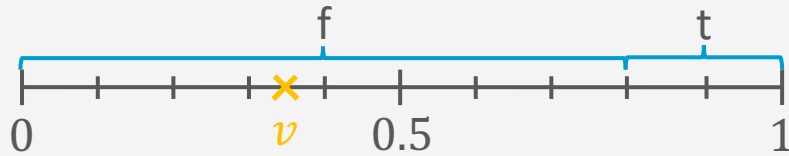


B	f_0	A	B	f_1	A	B	f_2
false	2	false	false	1	false	false	3
true	1	false	true	2	false	true	2
		true	false	3	true	false	4
		true	true	4	true	true	1

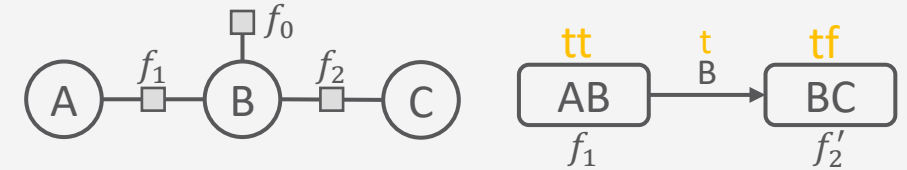
A	B	f_1^{BN}	B	C	f_2^{BN}
false	false	0.1	false	false	0.6
false	true	0.2	false	true	0.4
true	false	0.3	true	false	0.8
true	true	0.4	true	true	0.2

Example

- Ordering of AB, C
- Sample values for $BC \setminus B$ conditioned on $B = t$ from f_2^{BN}
 - With $B = t$, it is a distribution



- E.g., $v = 0.35 \rightarrow 0$
- Sample: $[t, t, f]$
 - Q weight: $Q([t, t, f]) = 0.4 \cdot 0.8$
 - $\tilde{P}$ weight: $\tilde{P}([t, t, f]) = 1 \cdot 4 \cdot 4$
 - $w([t, t, f]) = \frac{16}{0.32} = 50$



B	f_0	A	B	f_1	A	B	f_2
false	2	false	false	1	false	false	3
true	1	false	true	2	false	true	2
		true	false	3	true	false	4
		true	true	4	true	true	1

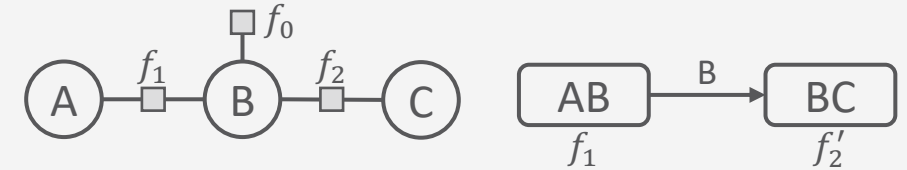
A	B	f_1^{BN}	B	C	f_2^{BN}
false	false	0.1	false	false	0.6
false	true	0.2	false	true	0.4
true	false	0.3	true	false	0.8
true	true	0.4	true	true	0.2

Example

- Ordering of AB, C
- Set of N samples $\mathbf{r}_i$ with weights $w(\mathbf{r}_i)$
 - E.g., sample $[t, t, f]$
 - $w([t, t, f]) = 50$
- Assume query for $P(C = \text{true})$
- Estimate

$$P(C) \approx \frac{\sum_{i=1}^N \mathbf{1}(\mathbf{r}_i, C = \text{true}) w(\mathbf{r}_i)}{\sum_{i=1}^N w(\mathbf{r}_i)}$$

$$\mathbf{1}(\mathbf{r}_i, C = \text{true}) = \begin{cases} 1 & \pi_C(\mathbf{r}_i) = \text{true} \\ 0 & \text{otherwise} \end{cases}$$



B	f_0	A	B	f_1	A	B	f_2
false	2	false	false	1	false	false	3
true	1	false	true	2	false	true	2
		true	false	3	true	false	4
		true	true	4	true	true	1

A	B	f_1^{BN}	B	C	f_2^{BN}
false	false	0.1	false	false	0.6
false	true	0.2	false	true	0.4
true	false	0.3	true	false	0.8
true	true	0.4	true	true	0.2

Lifted Importance Sampling (LIS)

- Consider an MLN $\Psi = \{(w_i, \psi_i)\}_{i=1}^n$
 - Probability of a world ω :

$$P_{\Psi}(\omega) = \frac{1}{Z_{\Psi}} \exp \left(\sum_{i=1}^n w_i n_i(\omega) \right)$$

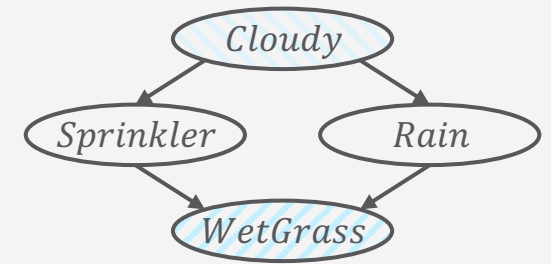
- Normalisation:

$$Z_{\Psi} = \sum_{\omega} \exp \left(\sum_{i=1}^n w_i n_i(\omega) \right)$$

- MLN in a *normal form*:
 - No constants in any formula
 - If any distinct atoms with the same predicate symbol have variables x, y in the same position, then x, y have the same domain
- Idea
 - Sample a value for one predicate
 - Value applies to all instances of predicate under the same evidence (group)
 - Use value to estimate quantities defined over the group

Problems with Importance Sampling

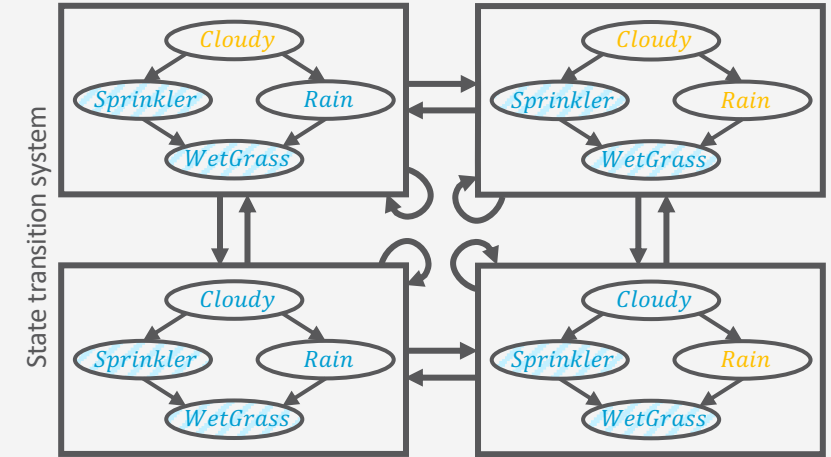
- Requires a reasonably fitting proposal distribution Q
 - Can be hard to construct/find if we deal with something other than directed models
- Cannot estimate distributions well for evidence in leaves
 - Independent of whether we deal with directed or undirected models
 - Consider two extreme cases in BNs (the easy model type)
 - All evidence at **roots**
 - Proposal distribution = posterior distribution
 - No weighting necessary (for all, $w = P(e)$)
 - All evidence at **leaves**
 - Proposal distribution = prior distribution
 - Correction purely by weights, yielding high variance
 - Will only work well if prior similar to posterior distribution; otherwise most samples are irrelevant, evidenced by a low weight



Markov Chain Monte Carlo (MCMC) Sampling

Gibbs Sampling, Metropolis-Hastings Sampling

Lifted Gibbs Sampling, Lifted Metropolis-Hastings Sampling

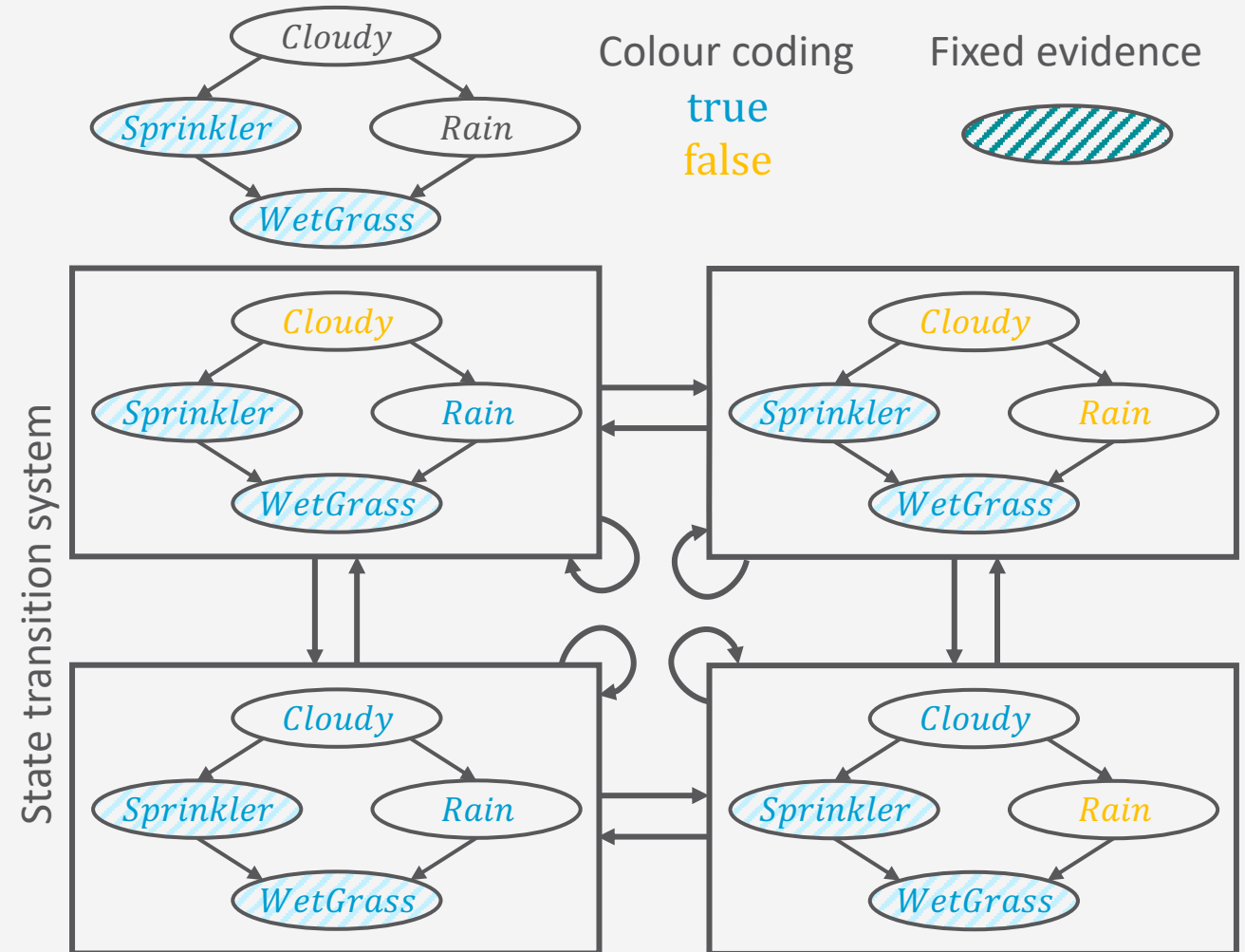


Markov Chain Monte Carlo (MCMC)

- Monte Carlo methods
 - Repeated random sampling to get to some numerical result
- Let us think of the model as being in a particular current state specifying a value for every variable
- MCMC generates each compound event by making a random change to the preceding event
 - Next state generated by randomly sampling a value for one non-evidence variable R_i conditioned on the current values of the variables in Markov blanket of R_i
 - Simplest form called Gibbs sampling, which the next slides build towards

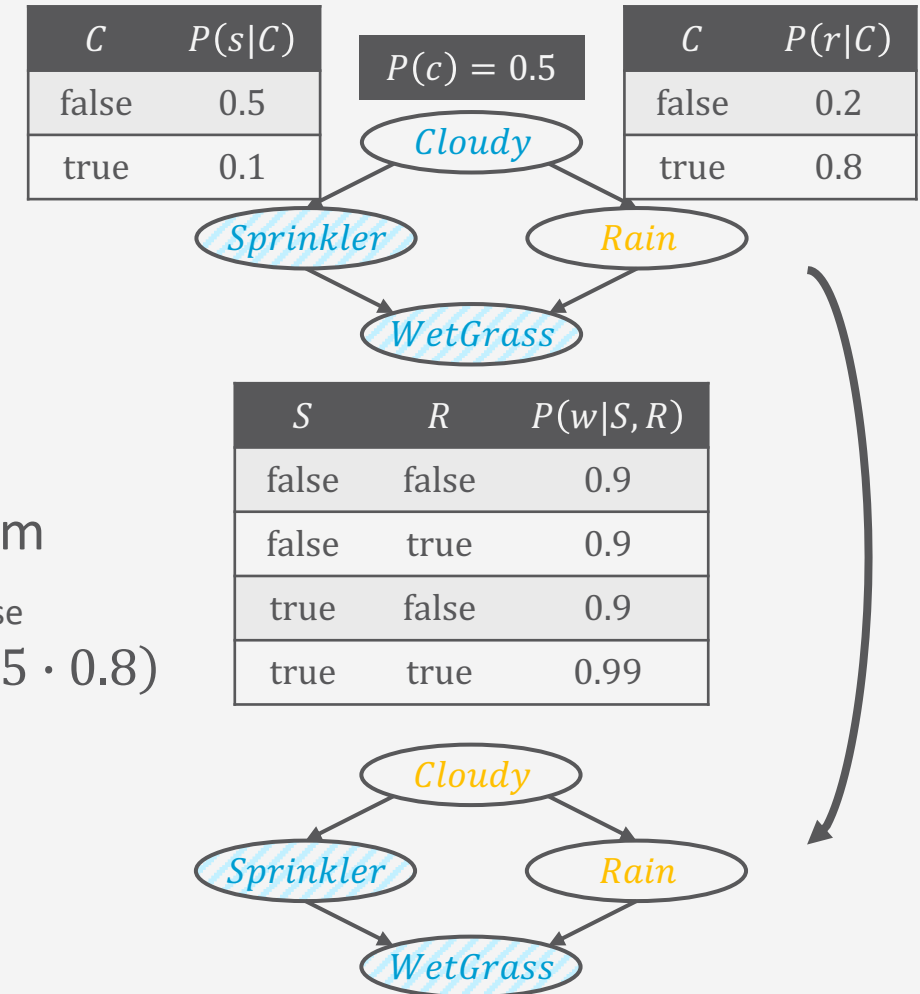
MCMC: Example

- Given $S = \text{true}$, $W = \text{true}$, four states (boxes) in state transition system
 - Four possible combinations of range values for remaining C, R
 - Arrows between states describe possible transitions
 - Probabilities from model
 - Leads to a (the Markov) chain of states
- Procedure:
Wander about for a while, average what you see



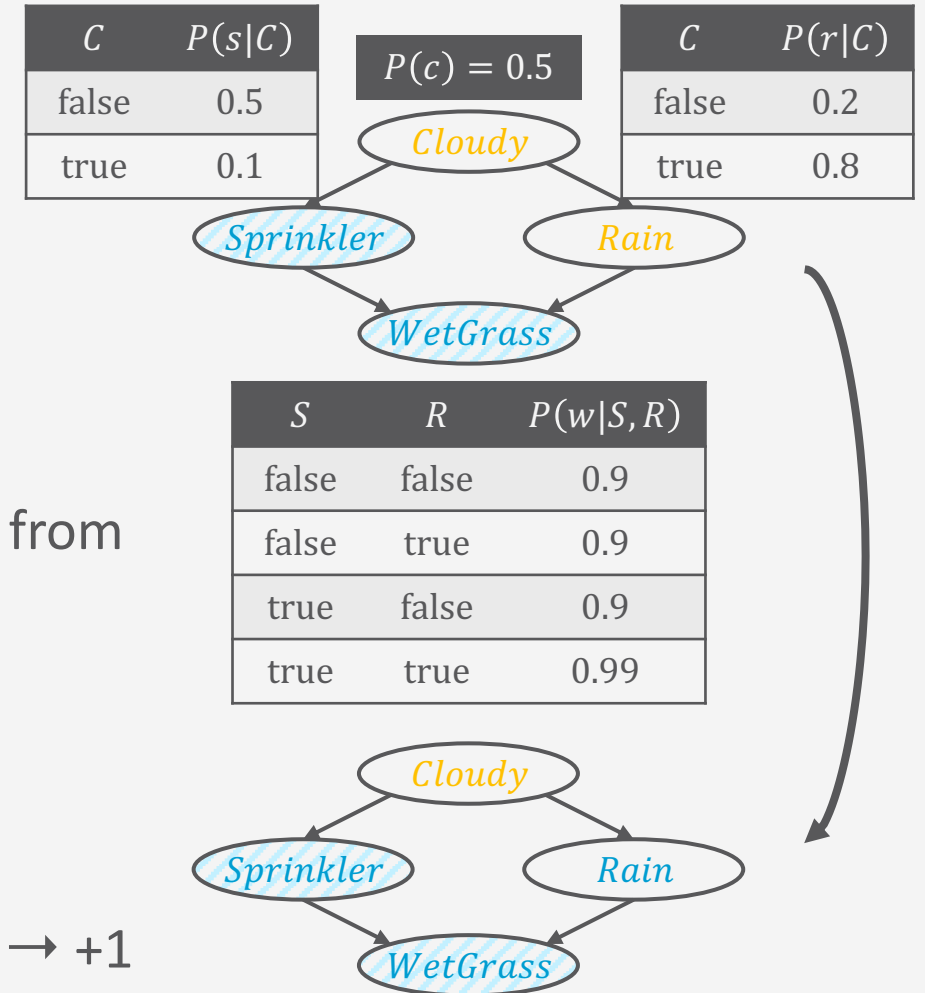
MCMC: Example

- $P(R|s, w)$?
 - Topological order (without evidence variables): (C, R)
 - Random choice of next variable to sample also possible
- Sampling (repeat N times)
 - Random initial state: $[c, s, \neg r, w]$
 - Sample C given current values of $MB(C) = \{S, R\}$, i.e., from $P(C|s, \neg r)$
 - $P(C|s, \neg r) = P(C)P(\neg r|C)P(s|C) = (0.5 \cdot 0.1 \cdot 0.2, 0.5 \cdot 0.5 \cdot 0.8) = (0.01, 0.2) = (0.05, 0.95)$
 - Suppose result is $\neg c$
 - New current state: $[\neg c, s, \neg r, w]$
 - Update count: $R = \text{false} \rightarrow +1$



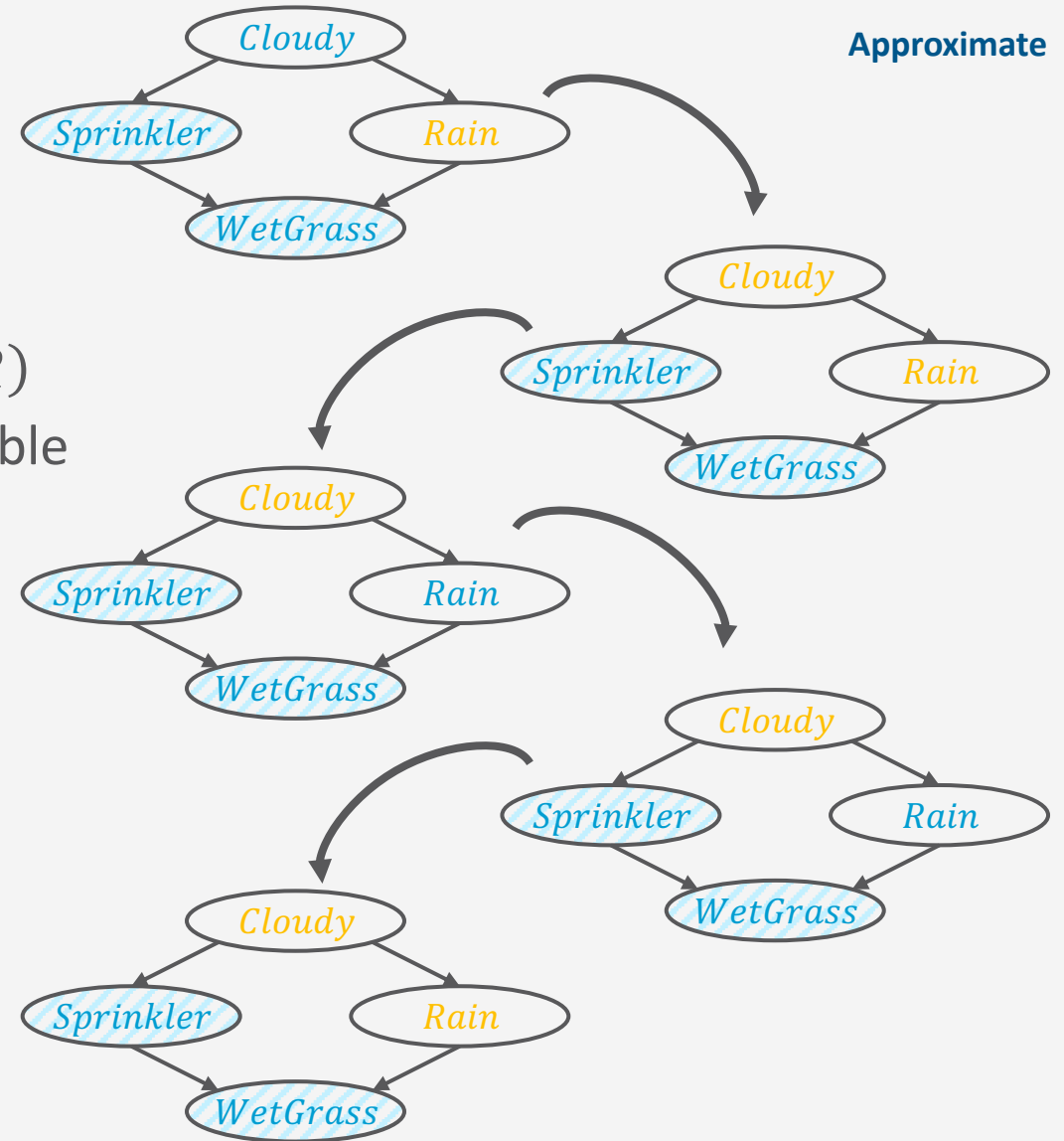
MCMC: Example

- $P(R|\text{shaded})$?
 - Topological order (without evidence variables): (C, R)
 - Random choice of next variable to sample also possible
- Sampling (repeat N times)
 - Current state: $[\neg c, \text{shaded}, \neg r, \text{shaded}]$
 - Sample R given current values of $MB(R) = \{C, S, W\}$, i.e., from $P(R|\neg c, s, w)$
 - Suppose result is r
 - New current state: $[\neg c, \text{shaded}, \underline{r}, \text{shaded}]$
 - Update count: $R = \text{true} \rightarrow +1$
 - Sample C from $P(C|s, r) \rightarrow [\neg c, \text{shaded}, r, \text{shaded}]$, $R = \text{true} \rightarrow +1$
 - Sample R from $P(R|\neg c, s, w) \rightarrow [\neg c, \text{shaded}, \neg r, \text{shaded}]$, $R = \text{false} \rightarrow +1$



MCMC: Example

- $P(R|\text{shaded})$?
 - Topological order (without evidence variables): (C, R)
 - Random choice of next variable to sample also possible
- Random initial state: $[c, \text{shaded}, \neg r, \text{shaded}]$
 - Next state: $[\neg c, \text{shaded}, \neg r, \text{shaded}]$
 - Next state: $[\neg c, \text{shaded}, \text{shaded}, r, \text{shaded}]$
 - Next state: $[\neg c, \text{shaded}, \text{shaded}, r, \text{shaded}]$
 - Next state: $[\neg c, \text{shaded}, \neg r, \text{shaded}]$
- Suppose that after $N = 80$ iterations, the process has visited 20 states with $R = \text{true}$ and 60 states with $R = \text{false}$; query result:
 Normalise $((20, 60)) = (0.25, 0.75)$



Gibbs Sampling

- Given a BN B , evidence e , and query terms R'
- State = current assignment $\mathbf{r}$ to all random variables $R = \text{rv}(B)$
 - Initially, e for $\text{rv}(e)$ (fixed) and random values $\mathbf{u}$ for all non-evidence random variables $U = \text{rv}(B) \setminus \text{rv}(e)$
- Generate the next state by sampling a value for non-evidence random variable U given its Markov blanket $\text{MB}(U)$ with assignments from the current state $\mathbf{r}$
 - Sample value u for U from $P(U \mid \pi_{\text{MB}(U)}(\mathbf{r}))$
 - Replace value of U in $\mathbf{r}$ with u
 - Increment counter for $\mathbf{r}'$ of R' occurring in $\mathbf{r}$, i.e., where $\mathbf{r}' = \pi_{R'}(\mathbf{r})$
 - Sample each variable according to some order or randomly, keep evidence fixed

Gibbs Sampling: Algorithm

Gibbs(R', e, B, N)

Vector N of length $|\text{ran}(R')|$, initially 0 ▷ stores counters for all $r' \in \text{ran}(R')$

Current state r consisting of e and random values u for $\text{rv}(B) \setminus \text{rv}(e)$

for $i = 1 \dots N$ **do**

for $U \in \text{rv}(B) \setminus \text{rv}(e)$ **do**

$u \leftarrow \text{Sample value for } U \text{ from } P(U \mid \pi_{\text{MB}(U)}(r))$

$r[U] \leftarrow u$ ▷ Replace value of U in r by u

$N[r'] \leftarrow N[r'] + 1$ with $r' = \pi_{R'}(r)$

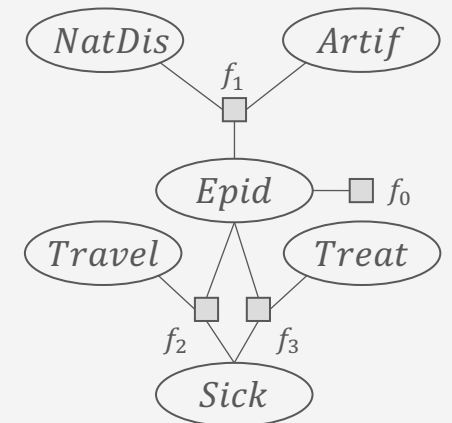
return Normalise(N)

Gibbs Sampling

Gibbs Sampling and Factor-based Models

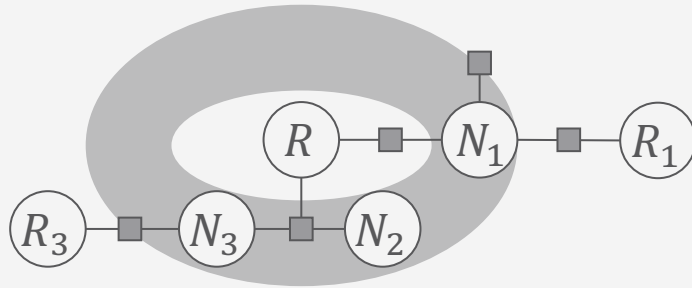
- Sample from $P(U | \pi_{\text{MB}(U)}(\mathbf{r}))$ in model F
- As U given $\text{MB}(U)$ independent of all other random variables in F and values for $\text{MB}(U)$ from current state $\mathbf{r}$ available:
Use **normalised product** $P(U, \pi_{\text{MB}(U)}(\mathbf{r}))$
of the factors F_U between U and $\text{MB}(U)$ with values $\pi_{\text{MB}(U)}(\mathbf{r})$
 - $F_U = \{f \mid f \in F, U \in \text{rv}(f)\}$
 - $P(U, \text{MB}(U)) = \frac{1}{Z} \prod_{f \in F_U} f$
- Example: State $[e, \neg n, \neg a, s, \neg tl, tt]$, query term Tl
 - Sample new value for, e.g., Tl from $P(Tl \mid \pi_{\text{MB}(Tl)}(\mathbf{r})) = P(Tl \mid e, s)$
 - $F_{Tl} = \{f_2\}$: normalise $\phi_2(Tl, e, s) \rightarrow P(Tl, e, s) = (0.25, 0.75)$
 - Suppose new value for Tl : **tl**, new state $[e, \neg n, \neg a, s, \mathbf{tl}, tt]$, $Tl = \text{true}$: +1

Travel	Epid	Sick	ϕ_2	P
false	false	false	20	
false	false	true	24	
false	true	false	5	
false	true	true	6	0.75
true	false	false	28	
true	false	true	8	
true	true	false	7	
true	true	true	2	0.25



Example with Two Factors

- Model



- Sample new value for R from $P(R|\pi_{MB(R)}(\mathbf{r})) = P(R|n_1, n_2, n_3)$
- Normalise $\phi(R, n_1) \cdot \phi(R, n_2, n_3) \rightarrow P(R, n_1, n_2, n_3)$
- Given $N_1 = \text{true}$, $N_2 = \text{true}$, $N_3 = \text{false}$
 - $f_{12} = \phi(R, \text{true}) \cdot \phi(R, \text{true}, \text{false}) = \phi(R)$
 - Sample new R value from $f'_{12} = \frac{1}{Z} f_{12}$

R	N_1	f_1
f	f	1
f	t	2
t	f	3
t	t	4

R	N_2	N_3	f_2
f	f	f	3
f	f	t	2
f	t	f	4
f	t	t	1
t	f	f	5
t	f	t	6
t	t	f	8
t	t	t	1

(MB values absorbed)

R	f_1
f	2
t	4

R	f_2
f	4
t	8

Product

R	f_{12}
f	8
t	32

Norm.

R	f'_{12}
f	0.2
t	0.8

Some Basics for MCMC

- A **Markov chain** consists of n states, plus an $n \times n$ transition matrix $\mathcal{T}$
 - At each step, we are in exactly one of the states
 - For $1 \leq i, j \leq n$, matrix entry $\mathcal{T}_{ij}$ tells us the relative frequency of j being the next state, given we are currently in state i



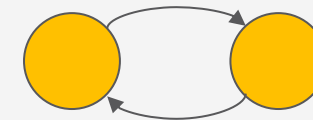
- Probability distribution, i.e.:

$$\sum_{j=1}^n \mathcal{T}_{ij} = 1$$

$\mathcal{T}_{ii} > 0$ ok (self loops)

Some Basics for MCMC

- Markov chain has to be **ergodic** for MCMC to work
- Markov chain is ergodic if
 - You have a path from any state to any other state (**irreducibility**)
 - No part of the system wanders off
 - Returns to states occur at irregular times (**aperiodicity**)
 - Periodicity: Returns to a state are only possible every $c > 1$ steps
- For any start state, after a finite transient time T_0 , the probability of being in any state at a fixed time $T > T_0$ is nonzero (**positive recurrence**)
 - Given a finite state space:
Positive recurrence follows from irreducibility



← Not ergodic
(even / odd)

Some Basics for MCMC

- **Ergodic theory**: about dynamical systems that are ergodic
 - System must be **measure-preserving**
 - Measure on a set: assign a number to each suitable subset of that set
 - Axioms of probability theory correspond to axioms of measure theory (*Kolmogorov axioms*)
 - Some ergodic theorems can be applied to probabilistic setting
- Some differences
 - In ergodic theory
 - irreducible + positive recurrent = ergodic and
 - irreducible + positive recurrent + aperiodic = mixing
 - Whereas in probability theory
 - irreducible + aperiodic + positive recurrent = ergodic

Kolmogorov axioms

1. Probability of an event is a non-negative real number
2. Assumption of unit measure: probabilities add up to 1
3. Assumption of σ -additivity: Probability of a set of disjoint events equals the sum over the individual probabilities (independence)

Some Basics for MCMC

- For any finite-state ergodic Markov chain, there is a unique **long-term visit rate** for each state
 - “*Steady-state*” or *stationary* distribution
 - *Stationarity*: Transition probabilities between states do not change over time
 - Over long time-period, each state visited in proportion to this rate
 - It does not matter where we start
- Reason why sampling works with a large enough sampling size

Some Basics for MCMC

- For any finite-state ergodic Markov chain, there is a unique **long-term visit rate** for each state
- Well-known application that you might have seen:
PageRank, original ranking principle of Google
 - Rank set of relevant web pages for a query according to the probabilities they have in the steady-state distribution (ranking is query independent)
 - Markov chain:
 - Web pages = states (i.e., being on one and not the others)
 - Arrows from one state/webpage to the next if outgoing link from one to the next
 - Transition model $\mathfrak{T}$: for each state, uniformly distributed over all outgoing links
 - Compute steady state distribution λ (as vector): λ has to fulfil $\lambda^T \mathfrak{T} = \lambda^T$
 - Eigenvector corresponding to eigenvalue 1

Stationary Distribution Formally

- A Markov chain is **regular** if there exists some number k such that, for every $\mathbf{r}, \mathbf{r}' \in \text{ran}(\mathbf{R})$, the probability of getting from $\mathbf{r}$ to $\mathbf{r}'$ in exactly k steps is > 0
 - For finite state spaces: Condition on regularity equivalent to condition on ergodicity
 - Sometimes easier to verify
 - In factor-based models:
If all potentials are strictly positive, then the Gibbs-sampling Markov chain is regular

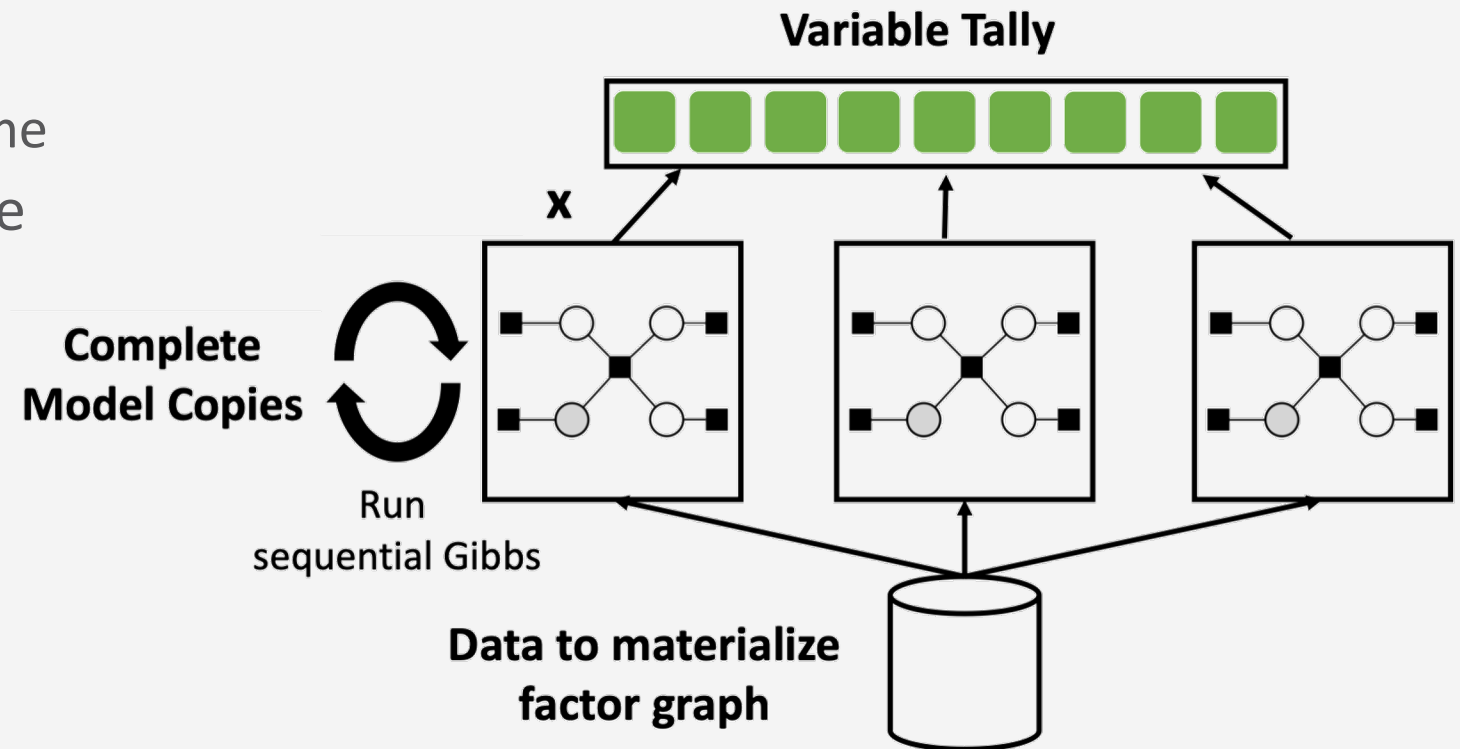
Stationary Distribution Formally

- Markov chain with transition model $\mathfrak{T}$ is **reversible** if there exists a unique distribution λ such that, for all $\mathbf{r}, \mathbf{r}' \in \text{ran}(\mathbf{R})$:
$$\lambda(\mathbf{r})\mathfrak{T}(\mathbf{r} \rightarrow \mathbf{r}') = \lambda(\mathbf{r}')\mathfrak{T}(\mathbf{r}' \rightarrow \mathbf{r})$$
- Equation is called *detailed balance*
 - Pick a starting state at random according to λ
 - Take a random transition from the chosen state according to $\mathfrak{T}$
- Asserts that, using this process, probability of a transition from $\mathbf{r} \rightarrow \mathbf{r}'$ is the same as probability of transition from $\mathbf{r}' \rightarrow \mathbf{r}$

If $\mathfrak{T}$ is regular and satisfies the detailed balance equation relative to λ , then λ is the unique stationary distribution of $\mathfrak{T}$.

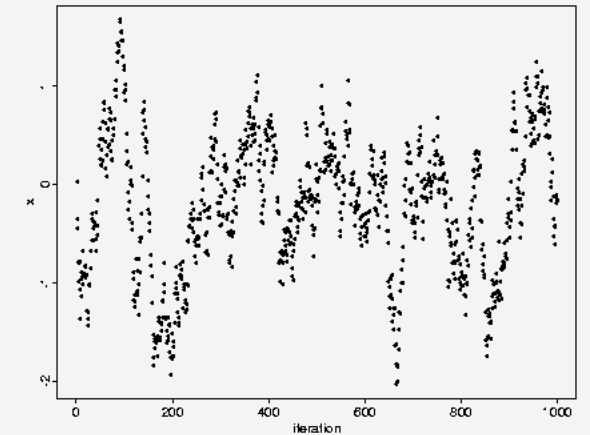
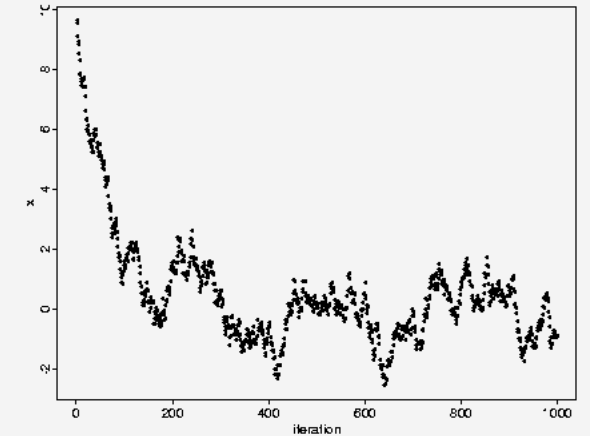
Parallelisation

- Run Gibbs independently on full copies of the same model
 - More samples in the same time
 - or
 - Same number of samples in less time
- Combine individual counters in one



Burn-in & Thinning

- Controversial techniques that each try to solve a problem
- Problem 1: Samples start at a random state that might be highly unlikely and skew the distribution
 - *Burn-in*/warm-up: Toss the first $N' < N$ samples
 - Alternatives
 - Start at highly likely state if known
 - Start at state that a previous run ended in
- Problem 2: As the next state depends on the previous one, the samples are no longer independent (autocorrelation)
 - *Thinning*/subsampling: Only take every k 'th sample
 - Does not really solve problem



A set of random variables following a mean-zero normal distribution; started at $x = 10$ and $x = 0$

Other Problems with Gibbs Sampling

- Only very local moves over the state space
 - One random variable at a time
 - In models with tightly correlated random variables, such moves can lead from highly likely states to states with very low probability
 - With a high probability of moving back to the high-probability state
 - Chain is unlikely to move away from such a state
 - Chain will mix slowly
- Consider chains that allow broader range of moves including larger steps
- Have to construct such a Markov chain with the same/desired stationary distribution

Metropolis-Hastings Algorithm (MH)

- Construct a Markov chain that is reversible with a particular stationary distribution λ
- Does not assume that we can generate next-state samples from a particular target distribution but uses the idea of a **proposal distribution**
 - Compare with importance sampling and its proposal distribution
 - Target distribution: next-state sampling distribution at a desired state
 - Sample from proposal distribution and correct for error
 - But: Do not keep track of importance weights
 - Are going to decay exponentially with number of transitions
 - Instead: Randomly choose whether to accept a proposed transition with a probability that corrects for the difference between proposal and target distribution

Proposal Distribution in MH

- **Proposal distribution** $\mathcal{I}^Q$ defines a transition model over state space $\text{ran}(\mathbf{R})$
 - For each state $\mathbf{r}$, $\mathcal{I}^Q$ defines a distribution over possible successor states in $\text{ran}(\mathbf{R})$, from which one randomly selects a candidate next state $\mathbf{r}'$
 - Either accept proposal and transition to $\mathbf{r}'$
 - Or reject proposal and stay at $\mathbf{r}$
 - For each pair of states $\mathbf{r}, \mathbf{r}'$, there exists an *acceptance probability* $\mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}')$
 - Actual transition model of Markov chain:

$$\mathcal{I}(\mathbf{r} \rightarrow \mathbf{r}') = \begin{cases} \mathcal{I}^Q(\mathbf{r} \rightarrow \mathbf{r}') \mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}') & \mathbf{r} \neq \mathbf{r}' \\ \mathcal{I}^Q(\mathbf{r} \rightarrow \mathbf{r}) + \sum_{\mathbf{r}' \neq \mathbf{r}} \mathcal{I}^Q(\mathbf{r} \rightarrow \mathbf{r}') (1 - \mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}')) & \text{sonst} \end{cases}$$

- Choice of proposal distribution arbitrary as long as it induces a regular chain

Acceptance Probabilities

- Given a proposal distribution $\mathfrak{T}^Q$, select acceptance probabilities $\mathfrak{A}$ to obtain desired stationary distribution λ
 - Detailed balance equation that has to hold

$$\lambda(\mathbf{r})\mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}')\mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}') \\ = \lambda(\mathbf{r}')\mathfrak{T}^Q(\mathbf{r}' \rightarrow \mathbf{r})\mathfrak{A}(\mathbf{r}' \rightarrow \mathbf{r})$$

- Set $\mathfrak{A}$ to be

$$\mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}') = \min \left[1, \frac{\lambda(\mathbf{r}')\mathfrak{T}^Q(\mathbf{r}' \rightarrow \mathbf{r})}{\lambda(\mathbf{r})\mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}')} \right]$$

Let $\mathfrak{T}^Q$ be any proposal distribution. Consider the Markov chain $\mathfrak{T}$ defined by

$$\mathfrak{T}(\mathbf{r} \rightarrow \mathbf{r}') \\ = \begin{cases} \mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}')\mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}') & \mathbf{r} \neq \mathbf{r}' \\ \mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}) + \sum_{\mathbf{r}' \neq \mathbf{r}} \mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}') (1 - \mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}')) & oth. \end{cases}$$

with

$$\mathfrak{A}(\mathbf{r} \rightarrow \mathbf{r}') = \min \left[1, \frac{\lambda(\mathbf{r}')\mathfrak{T}^Q(\mathbf{r}' \rightarrow \mathbf{r})}{\lambda(\mathbf{r})\mathfrak{T}^Q(\mathbf{r} \rightarrow \mathbf{r}')} \right].$$

If $\mathfrak{T}$ is regular, then it has the stationary distribution λ .

MH: Algorithm

- Follows the same procedure as Gibbs sampling **except**
 - Generate a new state $\mathbf{r}_i$ from proposal distribution $\mathcal{T}^Q$ instead of target distribution $\mathcal{T}$
 - Pick or discard $\mathbf{r}_i$ based on acceptance probability $\mathfrak{A}$

Gibbs($\mathbf{R}', \mathbf{e}, B, N$)

Vector N of length $|\text{ran}(\mathbf{R}')|$, initially 0

▸ stores counters for all $\mathbf{r}' \in \text{ran}(\mathbf{R}')$

Current state $\mathbf{r}$ consisting of $\mathbf{e}$ and random values $\mathbf{u}$ for $\text{rv}(B) \setminus \text{rv}(\mathbf{e})$

for $i = 1 \dots N$ **do**

for $U \in \text{rv}(B) \setminus \text{rv}(\mathbf{e})$ **do**

$u \leftarrow \text{Sample value for } U \text{ from } P(U \mid \pi_{\text{MB}(U)}(\mathbf{r}))$

$\mathbf{r}[U] \leftarrow u$

▸ Replace value of U in $\mathbf{r}$ by u

$N[\mathbf{r}'] \leftarrow N[\mathbf{r}'] + 1$ with $\mathbf{r}' = \pi_{\mathbf{R}'}(\mathbf{r})$

return Normalise(N)

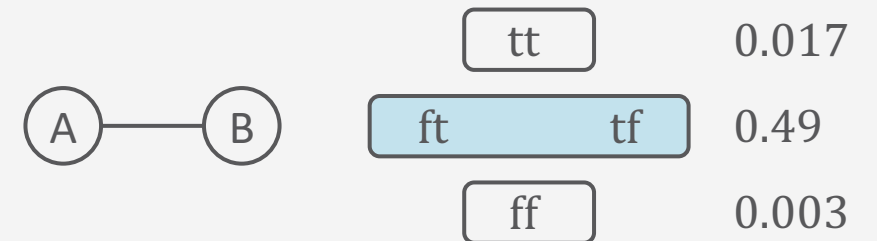
Gibbs Sampling

As far as I know at this point

Lifted MCMC

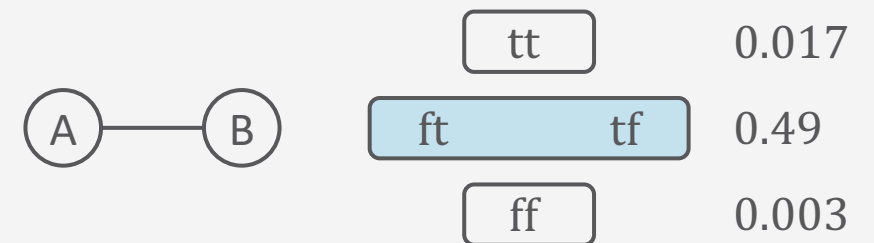
- **No direct transformation** of MCMC to lifted models
 - But: application of the lifting idea to Markov chains
- **Exchangeable** Boolean random variables $\mathcal{S}$
 - If for every assignment to $\mathcal{S}$, i.e., $\mathbf{s} \in \{f, t\}^k$, and every permutation g on $\{f, t\}^k$,

$$P(\mathcal{S} = \mathbf{s}) = P(\mathcal{S} = \mathbf{s}^g)$$
 - Example: Random variables that exhibit counting symmetry
 - Find these so-called **automorphism groups** using *colour passing* (forward pointer to next topic)
- Then, there are $k + 1$ **orbits** each containing random variable assignments
 - Here:
 Orbit $\approx$ equivalence class where elements within each class are mapped to the same probability



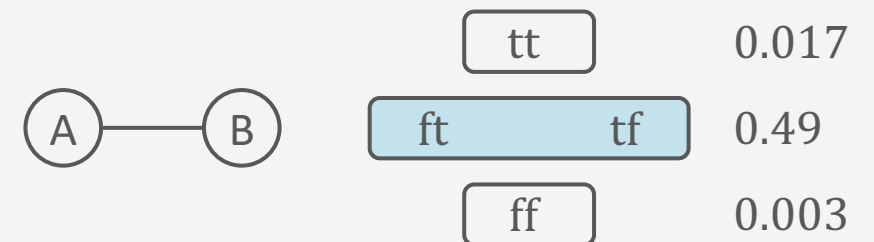
Why Do Orbits help?

- Example: Two Boolean random variables and a symmetric potential function
 - Probabilities of states ft and tf both 0.49
 - States ft and tf part of the same orbit
- Assume a standard Gibbs sampler is in state 10
 - Probability to transition to tt or ff is only 0.02 ($0.017 + 0.003$)
 - Cannot transition directly to state ft (two changes)
 - Chain is “stuck” in tf until it is able to move to tt or ff
- With orbital Gibbs sampler, intuitively, while it is “waiting” to move to one of the low probability states, it samples the two high probability states horizontally uniformly at random from the orbit $\{ft, tf\}$
 - Converges faster than standard Gibbs sampler
 - Can show analytically



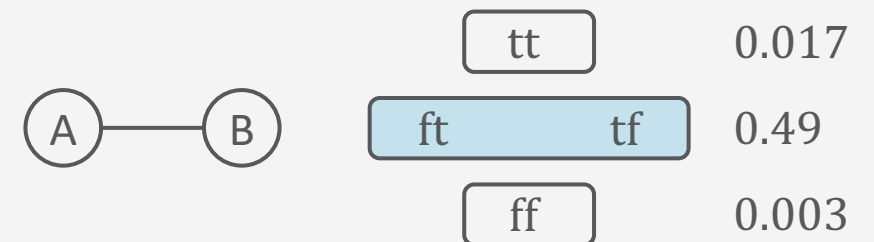
Orbital Markov Chain

- Assume standard Markov chain M' over state space $\text{ran}(\mathbf{R})$ with stationary distribution λ
- Let $\mathfrak{G}$ be an automorphism group on $(\text{ran}(\mathbf{R}), \lambda)$
- Orbital Markov chain M for M' performs:
 - Let $\mathbf{r}'$ be the state of M' at time t
 - Sample $\mathbf{r}$, the state of M at time t , uniformly at random from the orbit $\mathbf{r}'^{\mathfrak{G}}$ of $\mathbf{r}'$
- If M' is aperiodic/irreducible/reversible, then M also aperiodic/irreducible/reversible
- So, we can build a Gibbs sampler that converges to stationary distribution λ at least as fast or faster



Orbital Gibbs Sampling

- Two Markov chains,
 - One ordinary M'
 - One orbital M (based on symmetry groups)
- In each sampling iteration
 1. Run a step of traditional MCMC, chain M'
 - Select a random variable R uniformly at random
 - Sample a value for R based on the current states of M
 2. Sample the state of M uniformly at random from the orbit of the new state of R , i.e., select an equivalent state uniformly at random



Lifted MH

Given an orbital Metropolis chain A :

- Given symmetry group G (approx. symmetries)
- Orbit $\mathbf{x}^G$ contains all states approximately symmetric to $\mathbf{x}$
- In state $\mathbf{x}$
 1. Select $\mathbf{x}'$ uniformly at random from $\mathbf{x}^G$
 2. Move from $\mathbf{x}$ to $\mathbf{x}'$ with probability $\min \left\{ \frac{Pr(\mathbf{x})}{Pr(\mathbf{x}')} , 1 \right\}$
 3. Otherwise: stay in $\mathbf{x}$ (reject)
 4. Repeat

Account for evidence that may break symmetries, using, e.g., approximate symmetries
→ forward pointer to learning

and an ordinary (base) Markov chain B

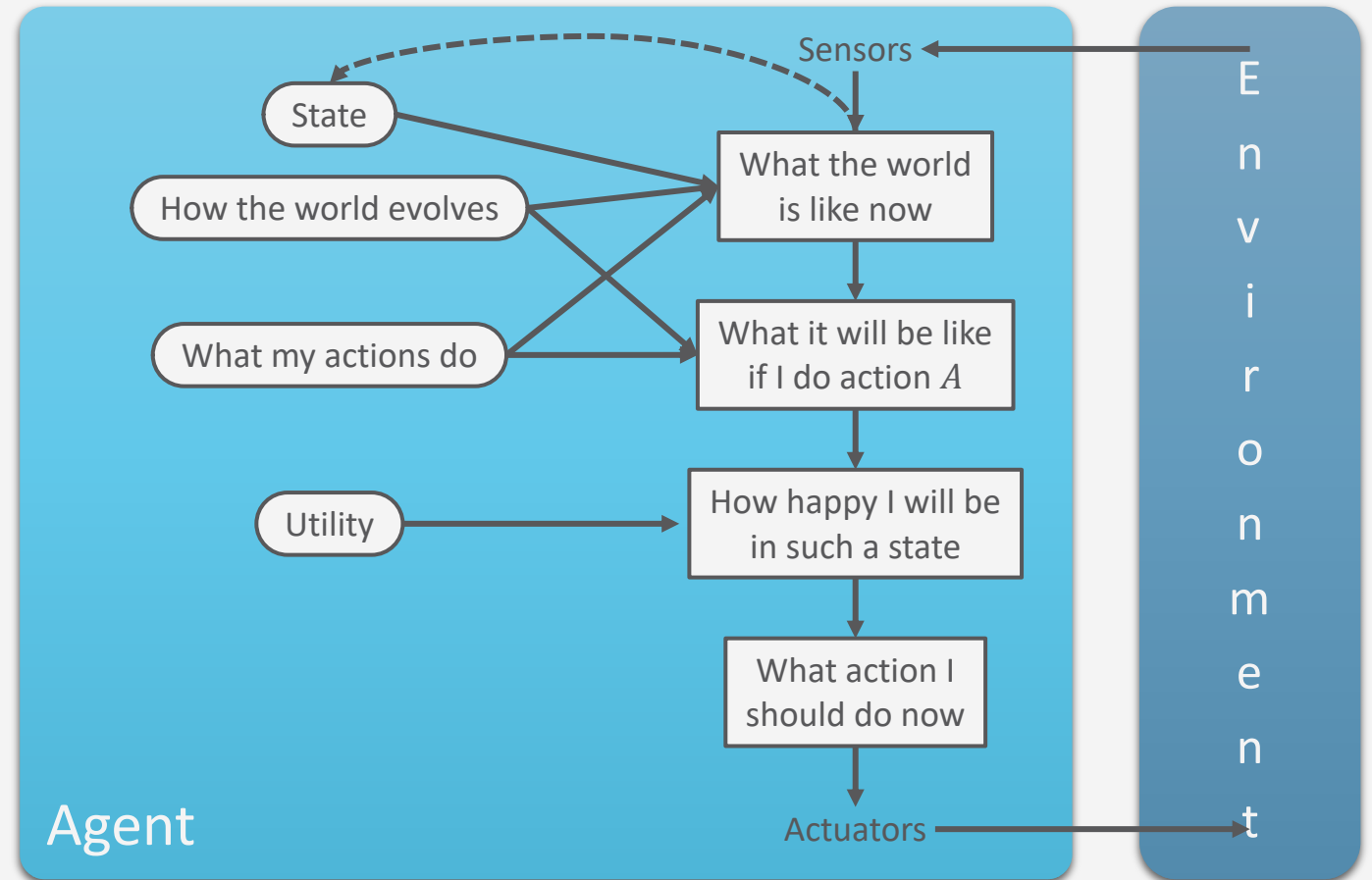
- With probability α follow B
- With $(1 - \alpha)$ follow A

Interim Summary

- Approximate inference based on sampling can lead to faster but approximate results
 - Goodness of approximation depends on the number of samples generated
- Direct sampling
 - Rejection sampling
 - Sample along graph structure, reject samples inconsistent with evidence
 - Importance sampling
 - Use proposal distribution for sampling, weight samples to correct the difference between proposal distribution and target distribution
 - Use domain knowledge about groups of indistinguishable instances to reduce variance
- MCMC sampling
 - Build a Markov chain and sample a new state based on the previous state
 - Find orbits for faster convergence

When to Choose Approximate or Exact Inference?

- Depends on:
 - Do you need exact results?
 - Can you get them in time / at all?
 - Can you get them numerically?
 - Can you sample from your model?
 - Can you get enough samples?
- What if we run both?
 - See what finishes first
 - Exact inference
 - Approximate inference with sufficient reliability
 - Give yourself a time horizon T



Agents: Monte Carlo vs. Las Vegas

- Agent has to work with available resources, requires an answer in a given time T
- **Monte Carlo** → Approximate inference (sampling)
 - The best possible but not necessarily correct result that could be generated in the given time
- **Las Vegas** → Exact inference
 - Either get the correct result in the given time or bust!
- Combine Monte Carlo & Las Vegas
 - While current time $t < T$
 - One thread works on exact inference, e.g., eliminate variables with LVE
 - One thread works on approximate inference, e.g., generate and count samples
 - If exact inference produces a result before t reaches T , break and return result
 - Otherwise: use result of approximate inference at T

Outline: 3. Inference

A. *Exact Inference*

- i. *(Weighted) Model Counting*
- ii. *Variable Elimination and the Like*
- iii. *Lifted Inference*

B. *Approximate Inference*

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling

→ Learning

Appendix

Lifted importance sampling

Lifted Importance Sampling (LIS)

- Consider estimating $Z_\psi = \sum_{\omega} \exp(\sum_{i=1}^n w_i n_i(\omega))$
 - Remember $E_{Q(R)}[w(R)] = Z$
- Then

$$Z_\psi = \sum_{\omega} \exp\left(\sum_{i=1}^n w_i n_i(\omega)\right) \frac{Q(\omega)}{Q(\omega)} = E_{Q(R)} \left[\frac{\exp(\sum_{i=1}^n w_i n_i(\omega))}{Q(\omega)} \right]$$

- Given N sampled worlds $\omega^{(t)}$, sampled independently from Q , then

$$Z \approx \hat{Z} = \frac{1}{N} \sum_{t=1}^N \frac{\exp(\sum_{i=1}^n w_i n_i(\omega^{(t)}))}{Q(\omega^{(t)})}$$

- LIS uses different lifting rules to handle instances as groups
 - Reduce variance for indistinguishable instances

LIS: Lifting Rules – Power Rule

- Given a normal MLN Ψ , a set of logical variables $\mathbf{x}$ is called a *decomposer* if it satisfies the following two conditions
 1. Every atom in Ψ contains exactly one variable from $\mathbf{x}$
 2. For any predicate symbol R , there exists a position s. t. variables from $\mathbf{x}$ only appear at that position in atoms of R
 - Any $x, y \in \mathbf{x}$ have the same domain because of normal form
- Given a decomposer $\mathbf{x}$ and any $x \in \mathbf{x}$, rewrite Z_Ψ as
$$Z_\Psi = \left(Z_{\Psi|_{\mathbf{x} \rightarrow x}} \right)^{|\text{dom}(\mathbf{x})|}$$
 - $\Psi|_{\mathbf{x} \rightarrow x}$ denoting that all occurrences of $\mathbf{x}$ are replaced with the same constant $x \in \text{dom}(\mathbf{x})$ and the resulting MLN is converted into a normal MLN

Compare DPGs in FO dtrees and set conjunctions in FO dDNNF circuits

LIS: Lifting Rules – Generalised Binomial Rule

Compare counting

- Given a normal MLN Ψ and a singleton atom $R(x)$ not involved in self-joins (does not appear more than once in same formula), rewrite Z_Ψ as

$$Z_\Psi = \sum_{j=0}^{|\text{dom}(x)|} \binom{|\text{dom}(x)|}{j} Z_{\Psi|r^j} w(j) 2^{p(j)}$$

- $\Psi|r^j$ denotes that in Ψ , truth values are assigned to $R(x)$ s.t. j instances are set to *true*; specifically
 - Ground all $R(x)$ and assign truth values to the groundings
 - Delete all formulas that evaluate to either *true* or *false*
 - Delete all groundings of $R(x)$
 - Convert the resulting MLN into a normal one
- $w(j)$ is the exponentiated sum of the weights of formulas that evaluate to *true*
- $p(j)$ is the number of ground atoms that are removed from the MLN as a result of removing formulas
 - Don't-care propositional atoms, which can be set to *true* or *false*
- Can be relaxed by not requiring singleton atoms but then no longer exact

LIS: Lifting Rules – Isolated Variable Rule

- For predicate symbol R of an MLN Ψ , a logical variable x at position m in its arguments is called *isolated*
 - if it is exclusive to R in all formulas containing R
- Let $\mathbf{x}$ denote the set of all isolated variables of R and let $\mathbf{y}$ denote the set of remaining variables in R
 - $\text{dom}(\mathbf{y})$ cartesian product of the domains of $\mathbf{y}$; $\mathbf{y}_i$ denotes the i^{th} element
- Then, estimate Z_Ψ as

$$Z_\Psi = Z_{\Psi|\mathbf{x}} w(R) 2^{p(R)} \prod_{i=1}^{|\text{dom}(\mathbf{y})|} \frac{\binom{|\text{dom}(\mathbf{x})|}{j_i}}{Q_i(j_i | j_1, \dots, j_{i-1})}$$

- $\Psi|\mathbf{x}$ an MLN obtained from Ψ by applying the following steps:
 1. For $i = 1$ to $|\text{dom}(\mathbf{y})|$, sample number j_i from a distribution $Q_i(j_i | j_1, \dots, j_{i-1})$ and set j_i arbitrarily selected groundings of $R(\mathbf{x}, \mathbf{y}_i)$ to *true* and the remaining to *false*,
 2. Delete all formulas that evaluate to either *true* or *false*
 3. Delete all groundings of R
 4. Convert the MLN to a normal one
- $w(R)$ exponentiated sum of the weights of formulas that evaluate to *true*
- $p(R)$ number of ground atoms that are removed from Ψ as a result of (2)

LIS: Algorithm (Z)

LIS tries to apply the power rule, followed by the generalised binomial rule, followed by the isolated variable rule. If all fail, then LIS grounds an atom and samples for the groundings.

```

function LIS( $\Psi$  – in normal form,  $Q$ ) returns an estimate of  $Z$ 
  if  $\Psi$  is empty then
    return 1
  if there exists a decomposer  $x$  then
    return  $(\text{LIS}(\Psi | x \rightarrow x, Q))^{|dom(x)|}$ 
  if there exists a singleton atom  $R(x)$  without self-joins then
    Use  $Q$  to sample an integer  $j \in \{0, \dots, |dom(x)|\}$ 
    return  $\frac{\text{LIS}(\Psi | r^j, Q) w(j) 2^{p(j)}}{Q(j)} \binom{|dom(x)|}{j}$ 
  if there exists isolated variables  $x$  in a predicate  $R$  then
    return  $\text{LIS}(\Psi | x, Q) w(R) 2^{p(R)} \prod_{j=1}^{|dom(y)|} \frac{\binom{|dom(x)|}{j_i}}{Q_i(j_i | j_1, \dots, j_{i-1})}$ 
  Choose an atom  $A$  and sample all of its groundings from  $Q$ 
  Let  $a$  be the sampled assignment
  return  $\frac{\text{LIS}(\Psi | a, Q) w(a) 2^{p(a)}}{Q(a)}$ 

```

LIS for Z

LIS: Constructing Q

- General ideas used
 - 4: disjoint parts
 - Handle independently
 - 8: choose an ordering for an atom
 - Assume parent-child relationship
- Lifting rules used for constructing Q
 - 2: power rule
 - Simplifies the MLN
 - 12: approximate generalised binomial rule
 - 13: isolated variable rule

Algorithm 2: Construct Proposal (CP)

Input: An MLN $\mathcal{M}$, an integer k and a set of atoms $\mathbf{R}$

Output: The structure of the proposal distribution Q

```

1 if  $\mathcal{M}$  is empty then return 1
2 if there exists a decomposer  $\mathbf{x}$  then
3    $\perp$  Let  $x \in \mathbf{x}$  and  $X \in \Delta_x$ . return  $\text{CP}(\mathcal{M}[X/\mathbf{x}], k, \mathbf{R})$ 
4 if  $\mathcal{M}$  can be decomposed into  $m$  MLNs  $\mathcal{M}_1, \dots, \mathcal{M}_m$  such
   that no two MLNs share any atoms then
5   for  $i = 1$  to  $m$  do
6      $\perp$   $\text{CP}(\mathcal{M}_i, k, \mathbf{R})$ 
7   return 1
8 Heuristically select an atom  $\mathbf{R}$  from  $\mathcal{M}$ 
9 Heuristically select  $k$  atoms from  $\mathbf{R}$  as parents of  $\mathbf{R}$ 
  // Construct Proposal over  $\mathbf{R}$ 
10 for every assignment to the groundings of  $\text{pa}(\mathbf{R})$  index by  $i$  do
11   if  $\mathbf{R}$  contains no isolated variables then
12      $\perp$  Use the approximate generalized binomial rule to
        construct  $Q_i(\mathbf{R})$ 
13   else
14      $\perp$  Use the isolated variables rule to construct  $Q_i(\mathbf{R})$ 
15 Add  $\mathbf{R}$  to  $\mathbf{R}$ 
16 Ground  $\mathbf{R}$  and then remove it from all formulas of  $\mathcal{M}$ 
17 return  $\text{CP}(\mathcal{M}, k, \mathbf{R})$ 
  
```