Bayesian network inference

- Given:
  - Query *variables:* **X**
  - *Evidence* (*observed*) variables: **E** = **e**
  - *Unobserved* variables: **Y**
- Goal: calculate some useful information about the query variables
  - Posterior P(**X**|**e**)
  - MAP estimate arg max$_{\mathbf{x}}$ P(**x**|**e**)
- Recall: inference via the full joint distribution

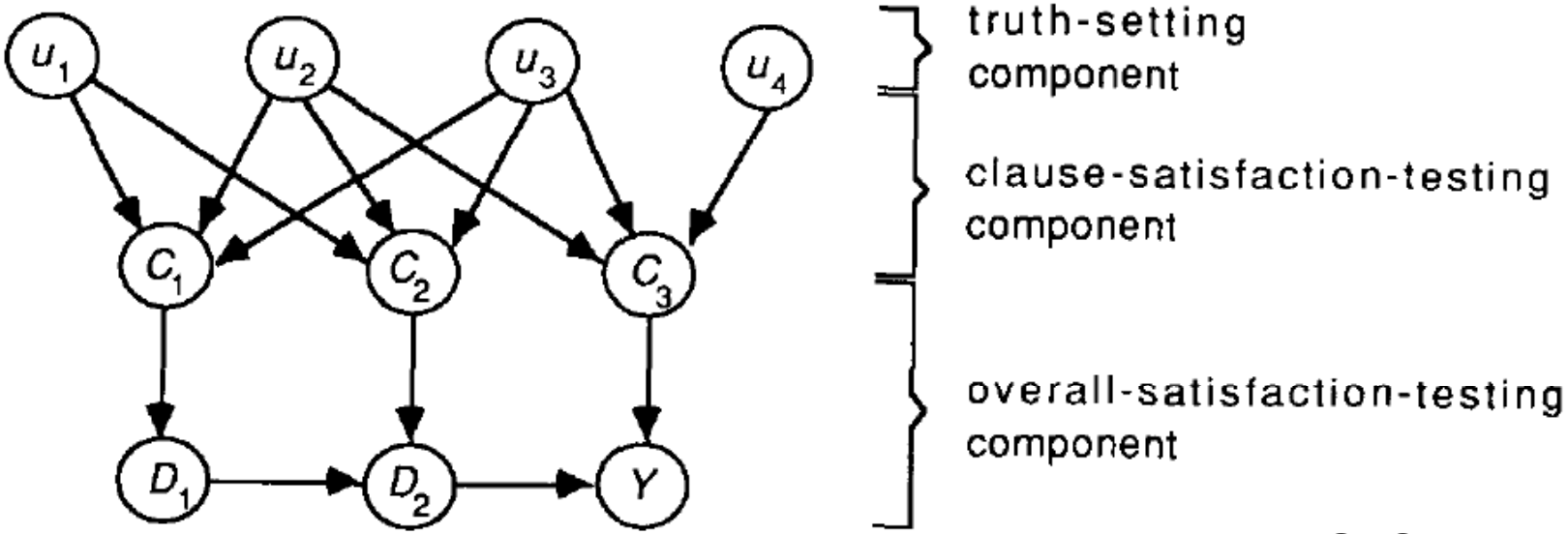
$$P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e}) = \frac{P(\boldsymbol{X}, \boldsymbol{e})}{P(\boldsymbol{e})} \propto \sum_{\boldsymbol{y}} P(\boldsymbol{X}, \boldsymbol{e}, \boldsymbol{y})$$

  - Since BN's can afford exponential savings in storage of joint distributions, can they afford similar savings for inference?

Bayesian network inference

- In full generality, NP-hard
  - More precisely, #P-hard: equivalent to counting satisfying assignments
- We can reduce satisfiability to Bayesian network inference
  - Decision problem: is P(Y) > 0?

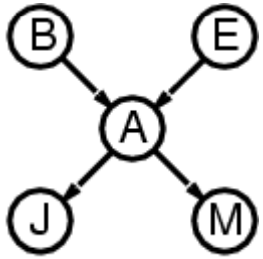
$$Y = \underbrace{(u_1 \vee u_2 \vee u_3)}_{C_1} \wedge \underbrace{(\neg u_1 \vee \neg u_2 \vee u_3)}_{C_2} \wedge \underbrace{(u_2 \vee \neg u_3 \vee u_4)}_{C_3}$$



G. Cooper, 1990

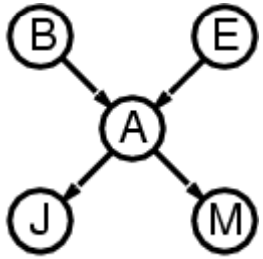
Inference example

- Query: P(B | j, m)



Inference example

- Query: P(B | j, m)



$$P(b \mid j,m) = \frac{P(b,j,m)}{P(j,m)} \propto \sum_{E=e}\sum_{A=a} P(b,e,a,j,m)$$

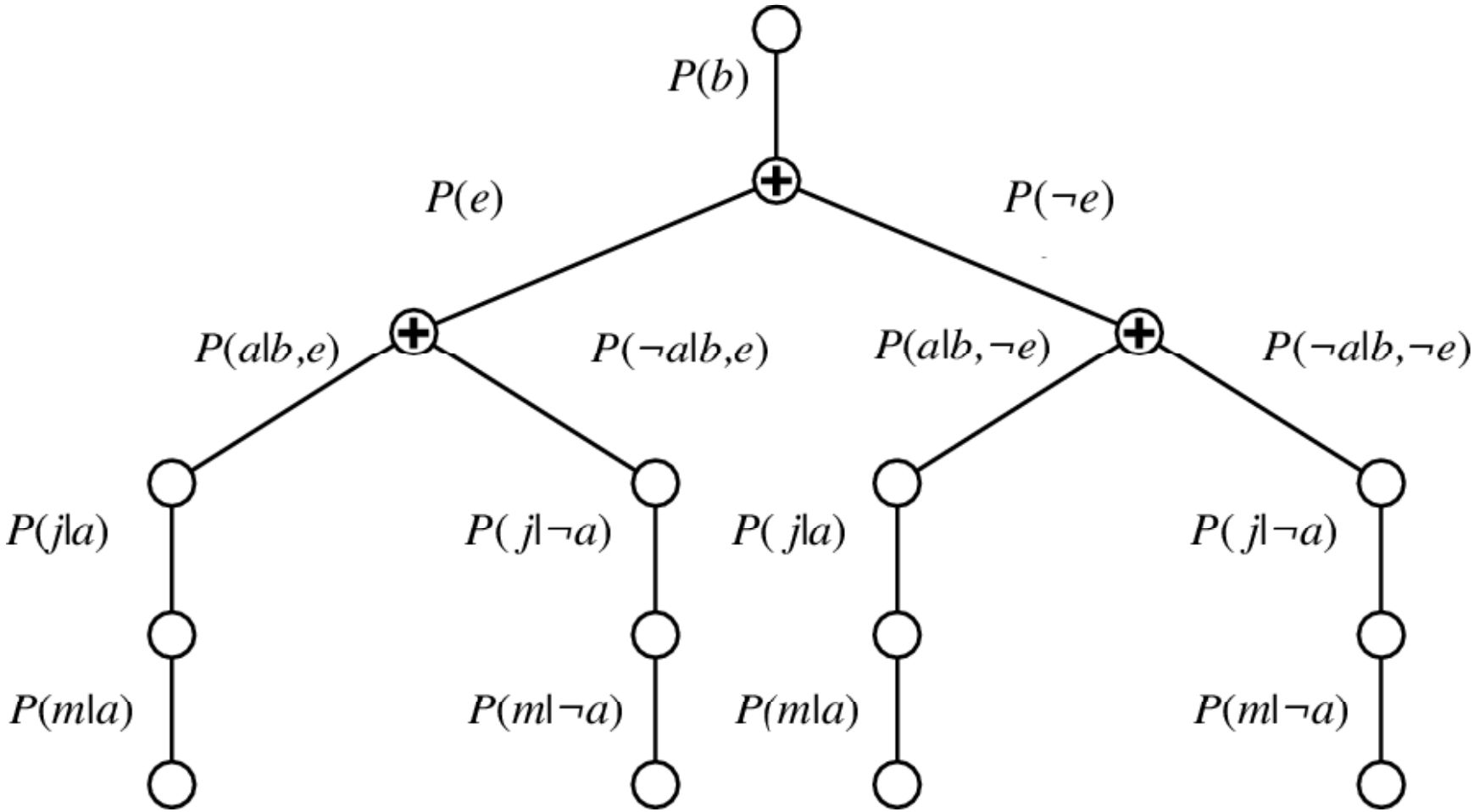
$$= \sum_{E=e}\sum_{A=a} P(b)P(e)P(a \mid b,e)P(j \mid a)P(m \mid a)$$

$$= P(b)\sum_{E=e} P(e)\sum_{A=a} P(a \mid b,e)P(j \mid a)P(m \mid a)$$

- Are we doing any unnecessary work?

Inference example

$$P(b \mid j,m) \propto P(b)\sum_{E=e} P(e)\sum_{A=a} P(a \mid b,e)P(j \mid a)P(m \mid a)$$

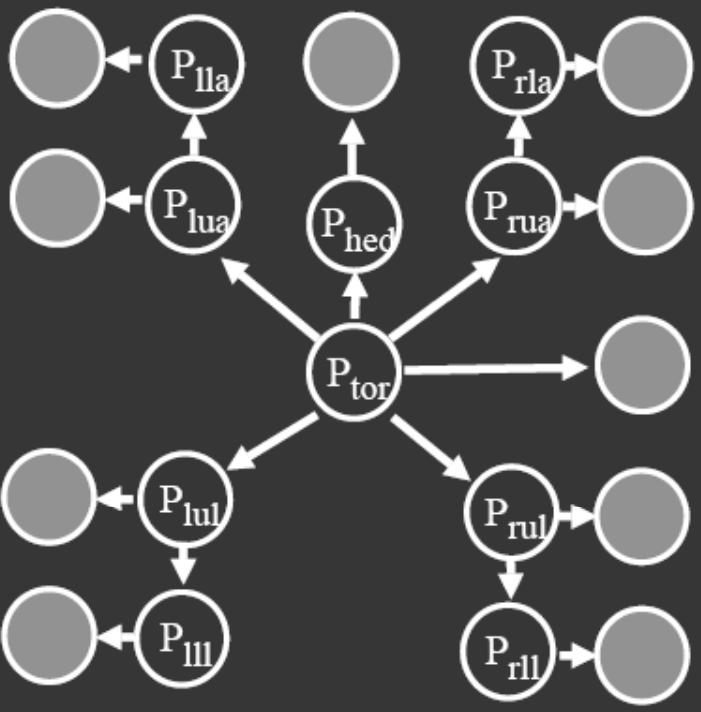


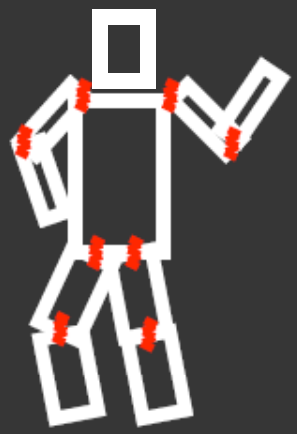
Exact inference

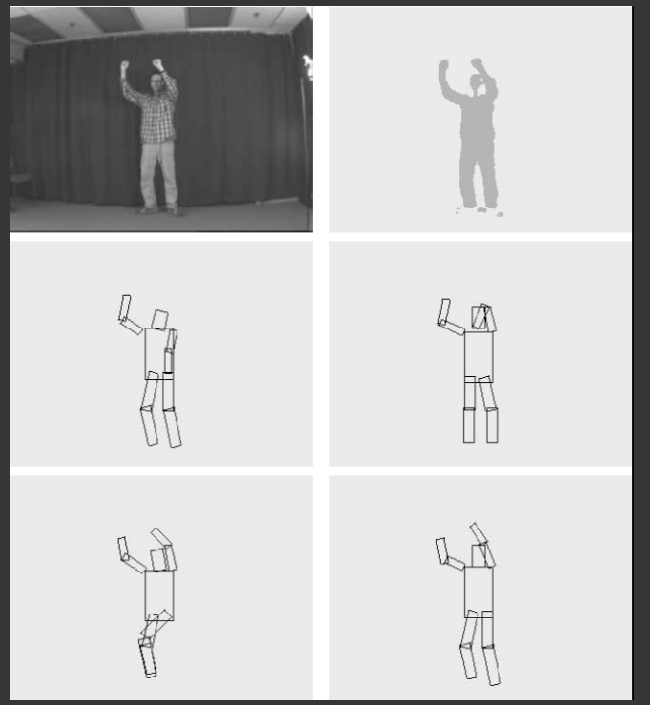
- Basic idea: compute the results of sub-expressions in a bottom-up way and cache them for later use
  - Form of dynamic programming
- Has polynomial time and space complexity for *polytrees*
  - Polytree: at most one undirected path between any two nodes

Pictorial structure model

Fischler and Elschlager(73), Felzenszwalb and Huttenlocher(00)







$$\Pr(P_{tor}, P_{arm}, \ldots | Im) \propto \prod_{i,j} \Pr(P_i | P_j) \prod_i \Pr(\mathrm{Im}(P_i))$$

part geometry

part appearance

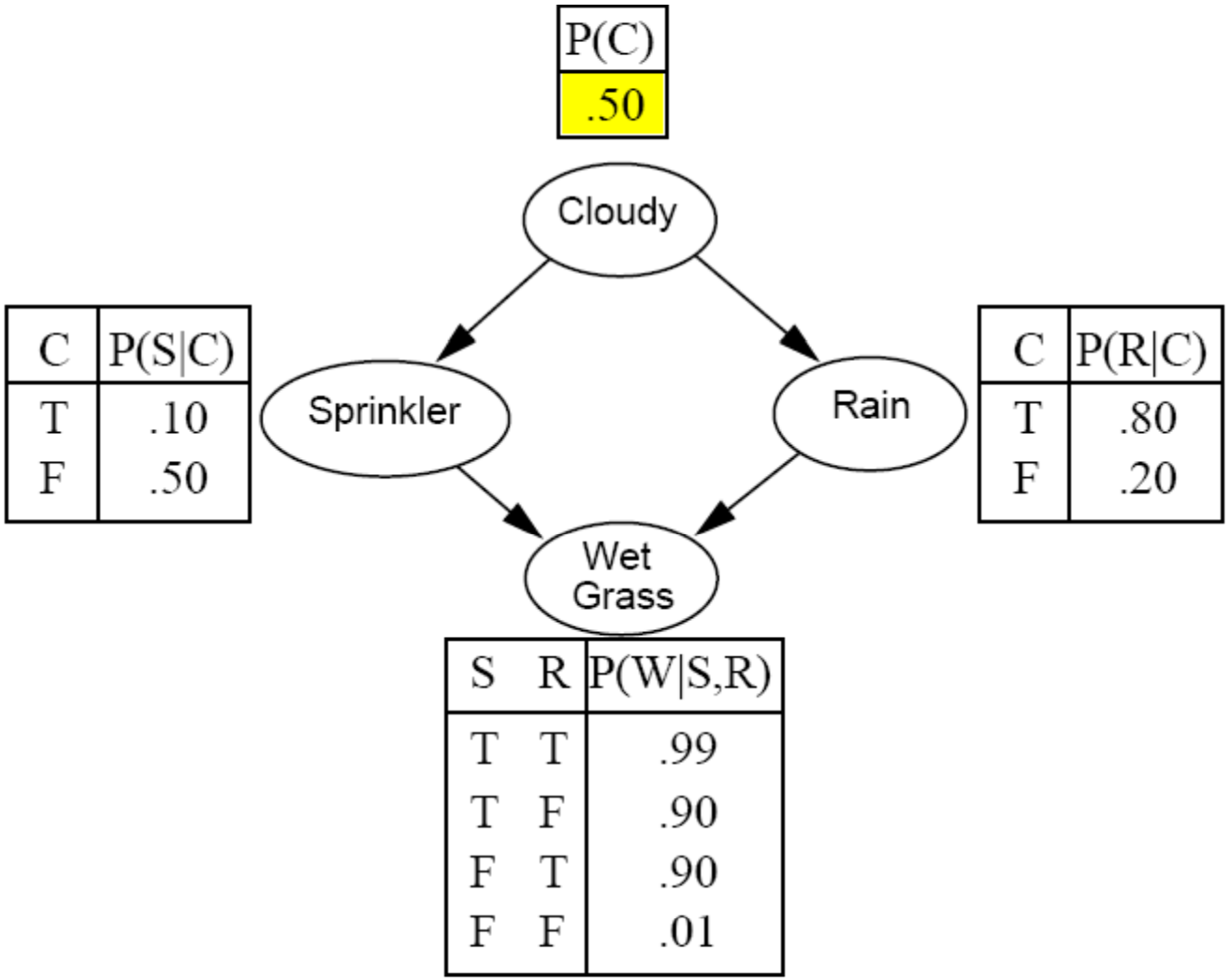
Review: Bayesian network inference

- In general, harder than satisfiability
- Efficient inference via dynamic programming is possible for *polytrees*
- In other practical cases, must resort to approximate methods

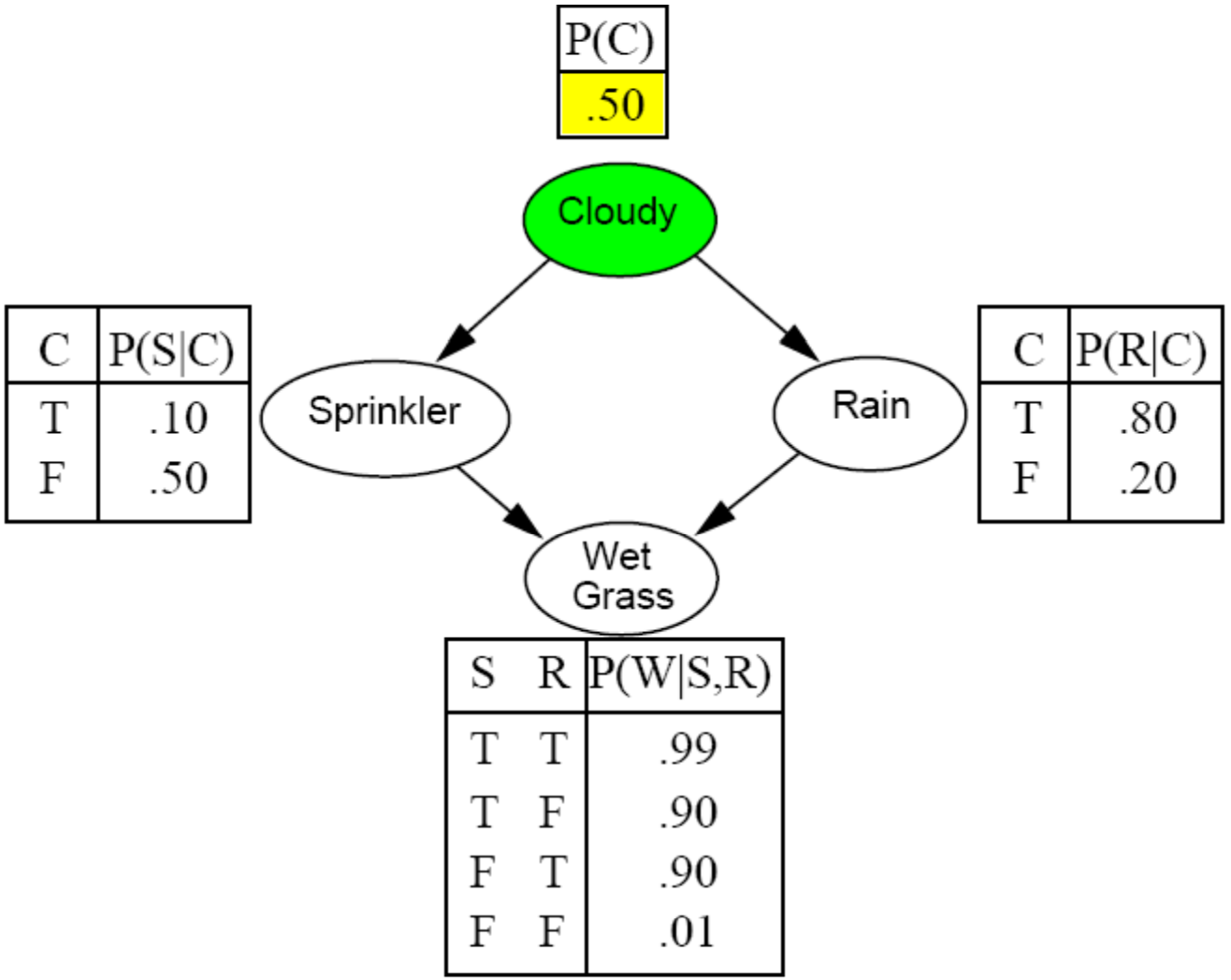
Approximate inference: Sampling

- A Bayesian network is a *generative model*
  - Allows us to efficiently generate samples from the joint distribution
- Algorithm for sampling the joint distribution:
  - While not all variables are sampled:
    - Pick a variable that is not yet sampled, but whose parents are sampled
    - Draw its value from P(X | parents(X))

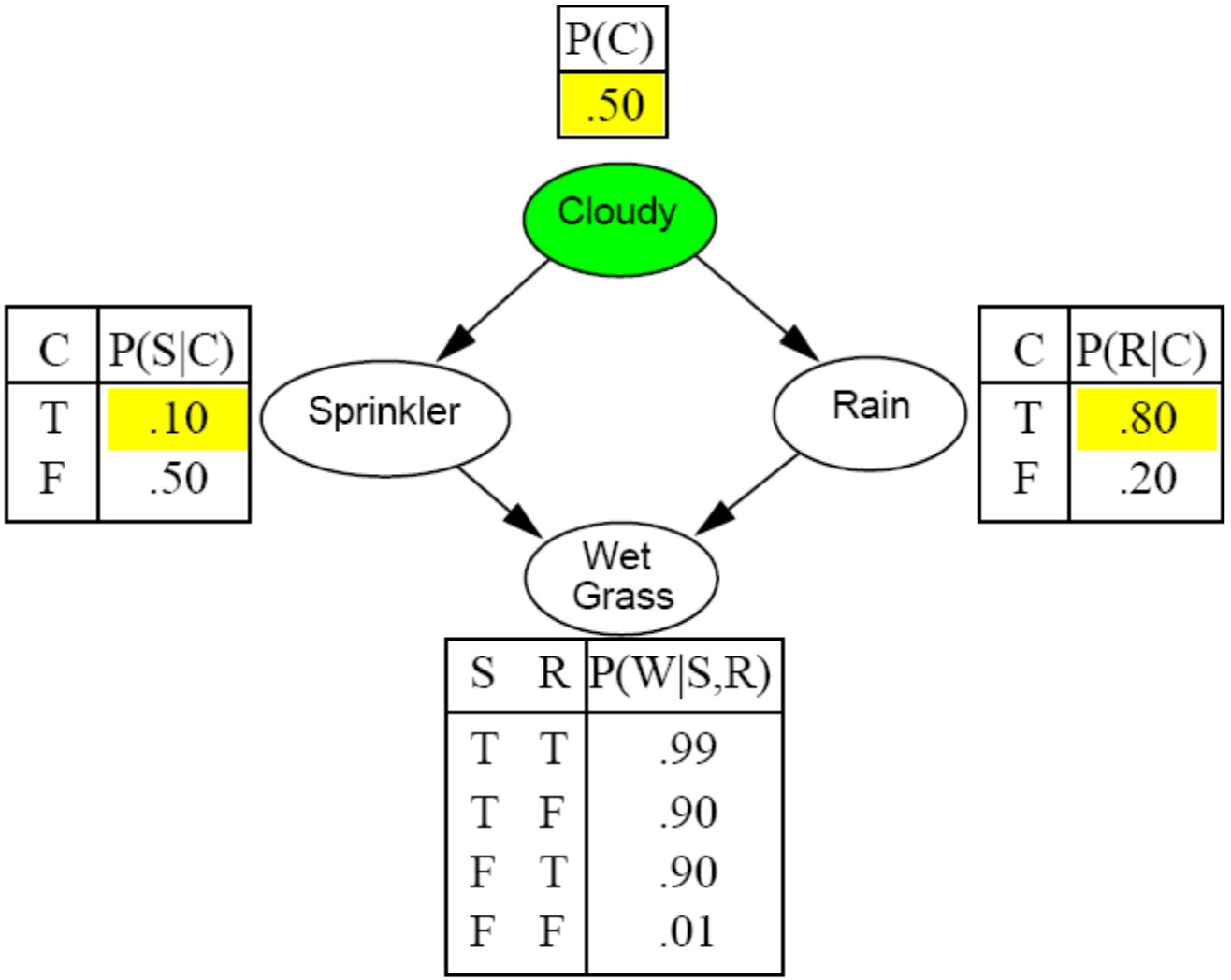
Example of sampling from the joint distribution



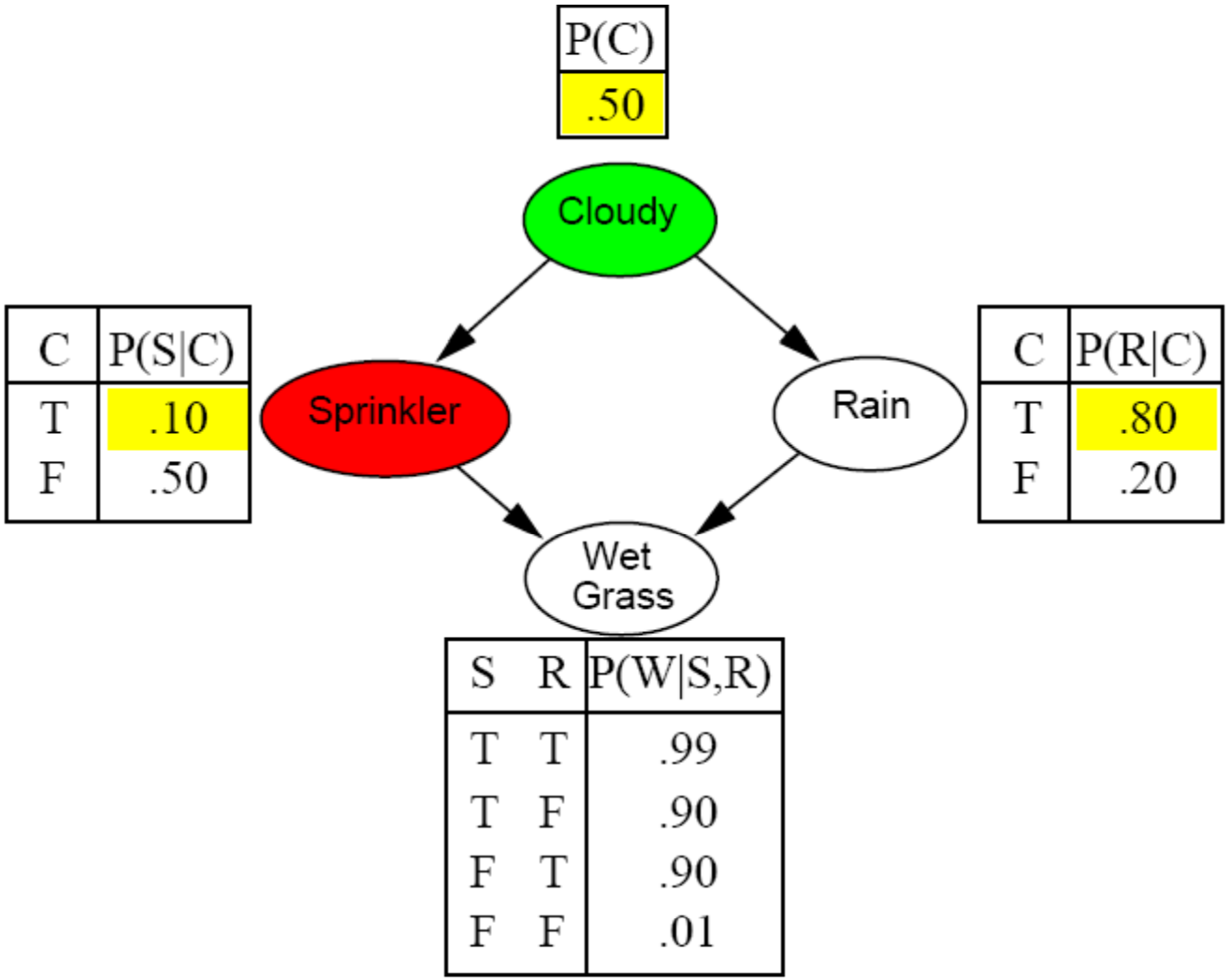
Example of sampling from the joint distribution



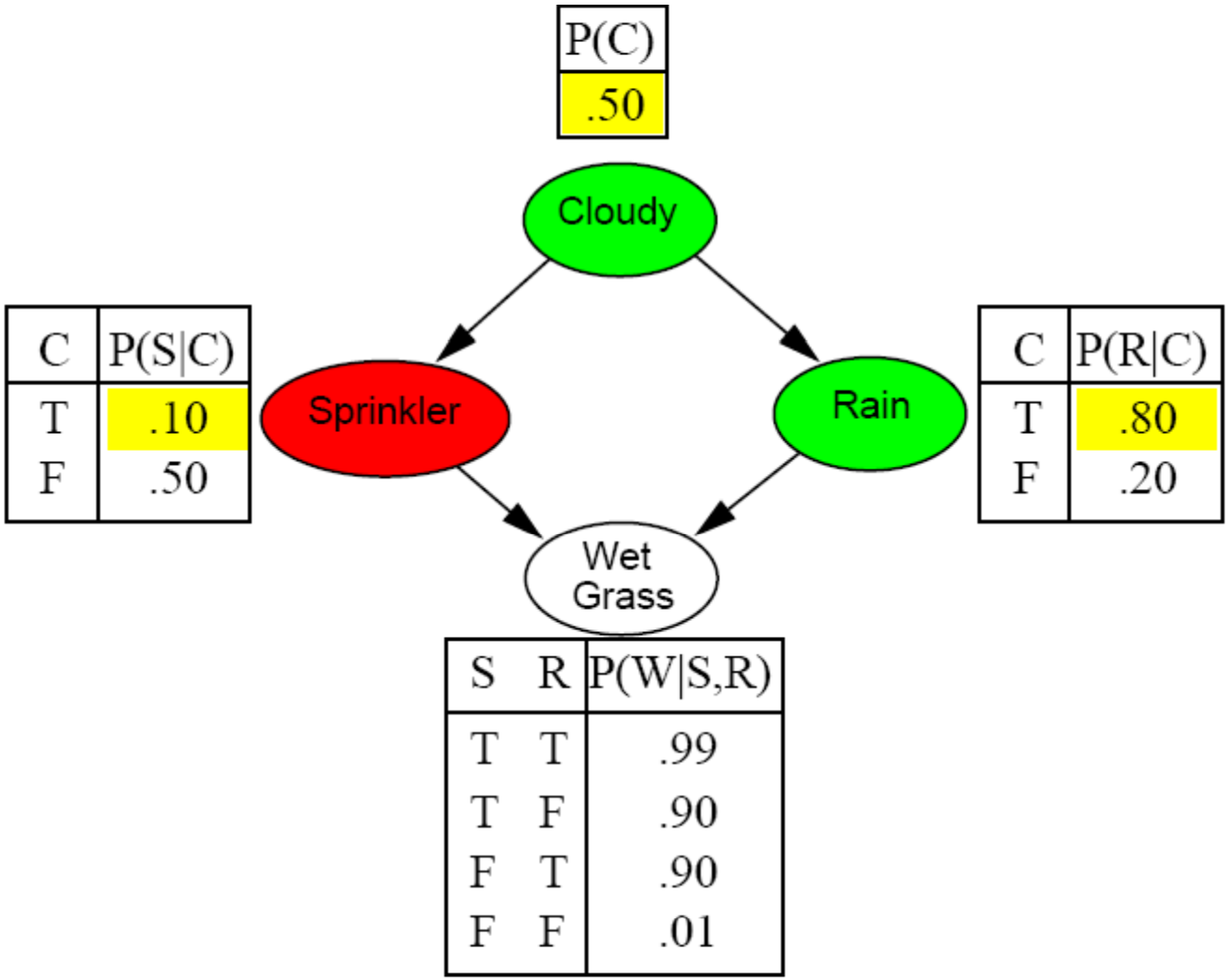
Example of sampling from the joint distribution



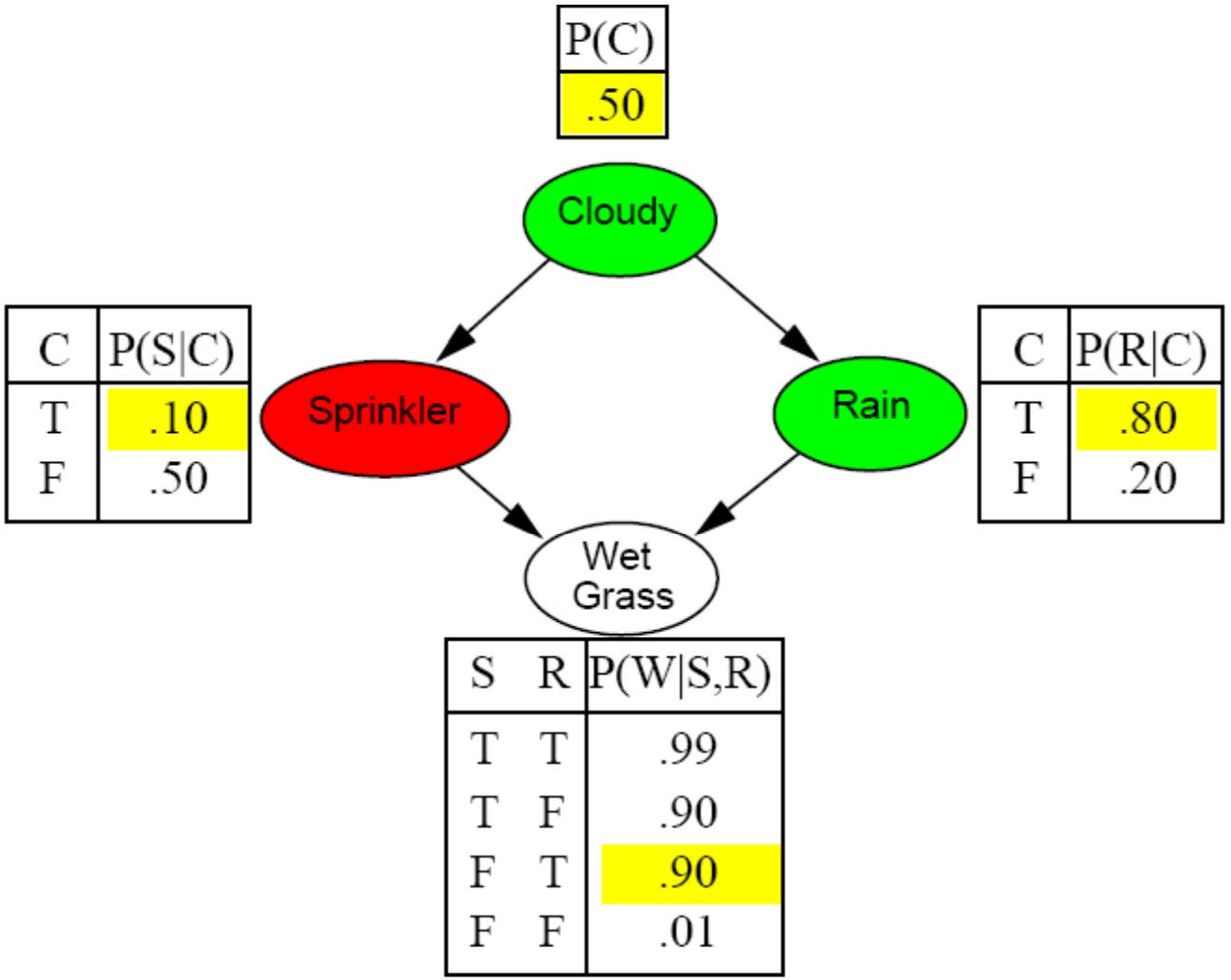
Example of sampling from the joint distribution



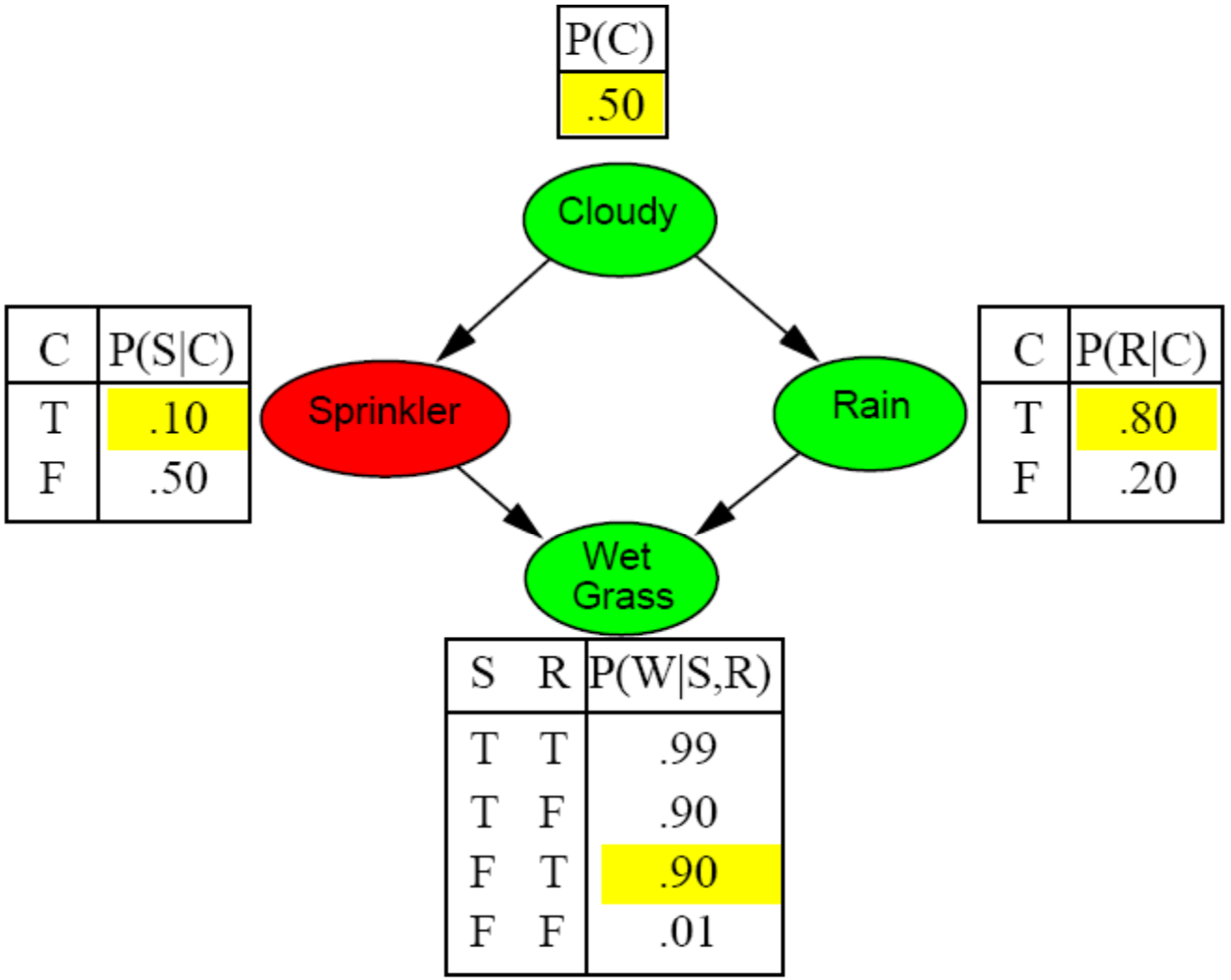
Example of sampling from the joint distribution



Example of sampling from the joint distribution



Example of sampling from the joint distribution



Inference via sampling

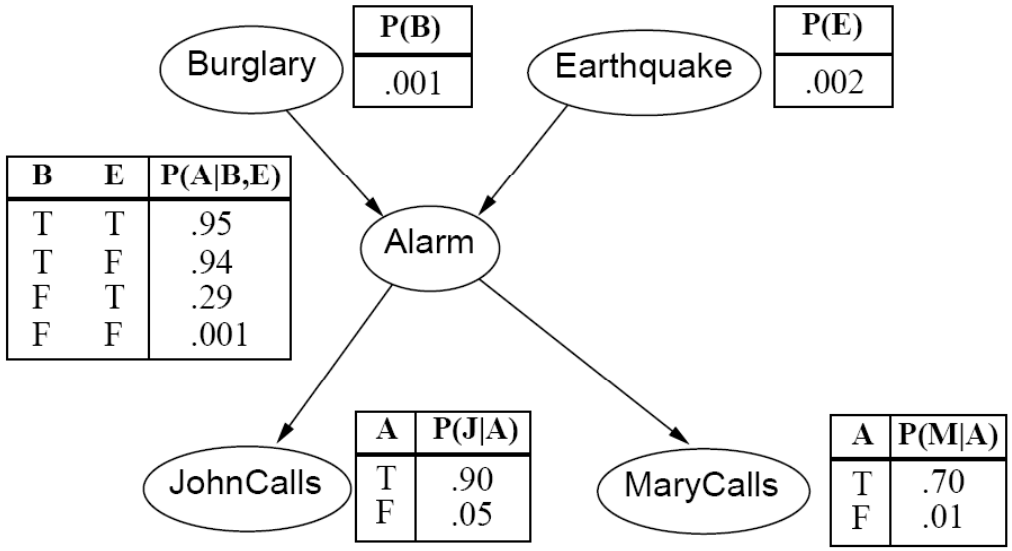
- Suppose we drew *N* samples from the joint distribution
- How do we compute P(**X** = **x** | **e**)?

$$P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{e}) = \frac{P(\boldsymbol{x}, \boldsymbol{e})}{P(\boldsymbol{e})} \approx \frac{\#\text{ of times } \boldsymbol{x} \text{ and } \boldsymbol{e} \text{ happen} / N}{\#\text{ of times } \boldsymbol{e} \text{ happens} / N}$$

- **Rejection sampling:** to compute P(**X** = **x** | **e**), keep only the samples in which **e** happens and find in what proportion of them **x** also happens

Inference via sampling

- **Rejection sampling:** to compute P(**X** = **x** | **e**), keep only the samples in which **e** happens and find in what proportion of them **x** also happens
- What if **e** is a rare event?
  - Example: burglary $\wedge$ earthquake
  - Rejection sampling ends up throwing away most of the samples



Inference via sampling

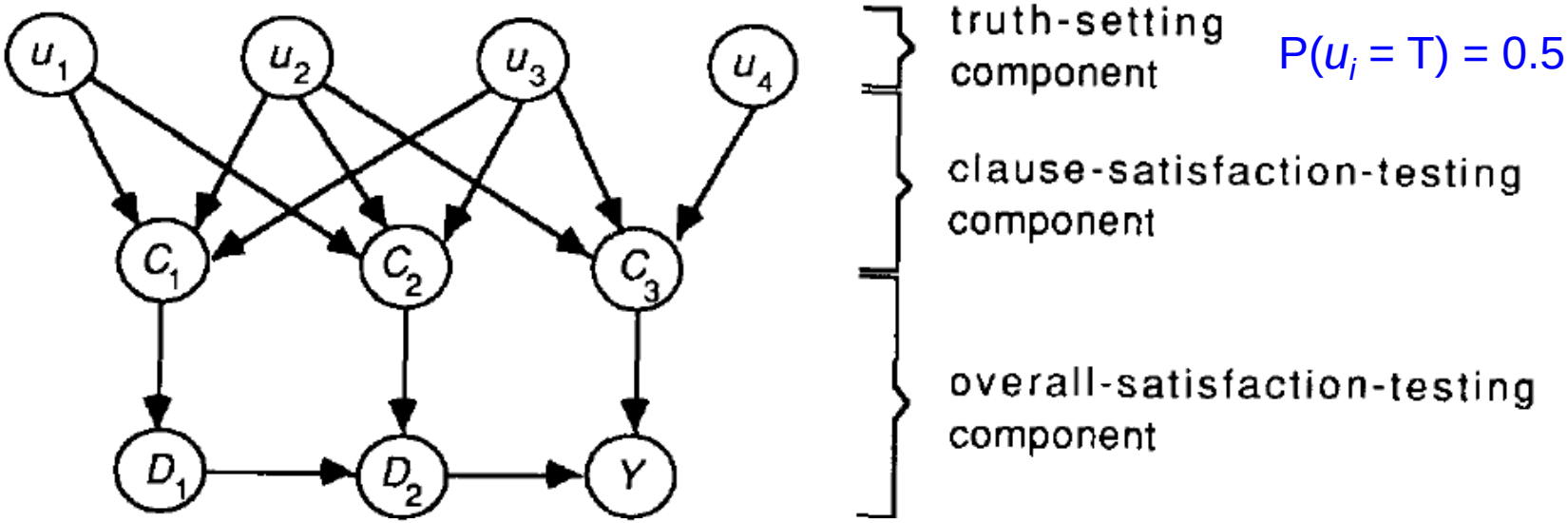
- **Rejection sampling:** to compute $P(\mathbf{X} = \mathbf{x} \mid \mathbf{e})$, keep only the samples in which **e** happens and find in what proportion of them **x** also happens
- What if **e** is a rare event?
  - Example: burglary $\wedge$ earthquake
  - Rejection sampling ends up throwing away most of the samples
- Likelihood weighting
  - Sample from $P(\mathbf{X} = \mathbf{x} \mid \mathbf{e})$, but weight each sample by $P(\mathbf{e})$

Inference via sampling: Summary

- Use the Bayesian network to generate samples from the joint distribution
- Approximate any desired conditional or marginal probability by empirical frequencies
  - This approach is **consistent**: in the limit of infinitely many samples, frequencies converge to probabilities
  - No free lunch: to get a good approximate of the desired probability, you may need an exponential number of samples anyway

Example: Solving the satisfiability problem by sampling

$$Y = \underbrace{(u_1 \vee u_2 \vee u_3)}_{C_1} \wedge \underbrace{(\neg u_1 \vee \neg u_2 \vee u_3)}_{C_2} \wedge \underbrace{(u_2 \vee \neg u_3 \vee u_4)}_{C_3}$$



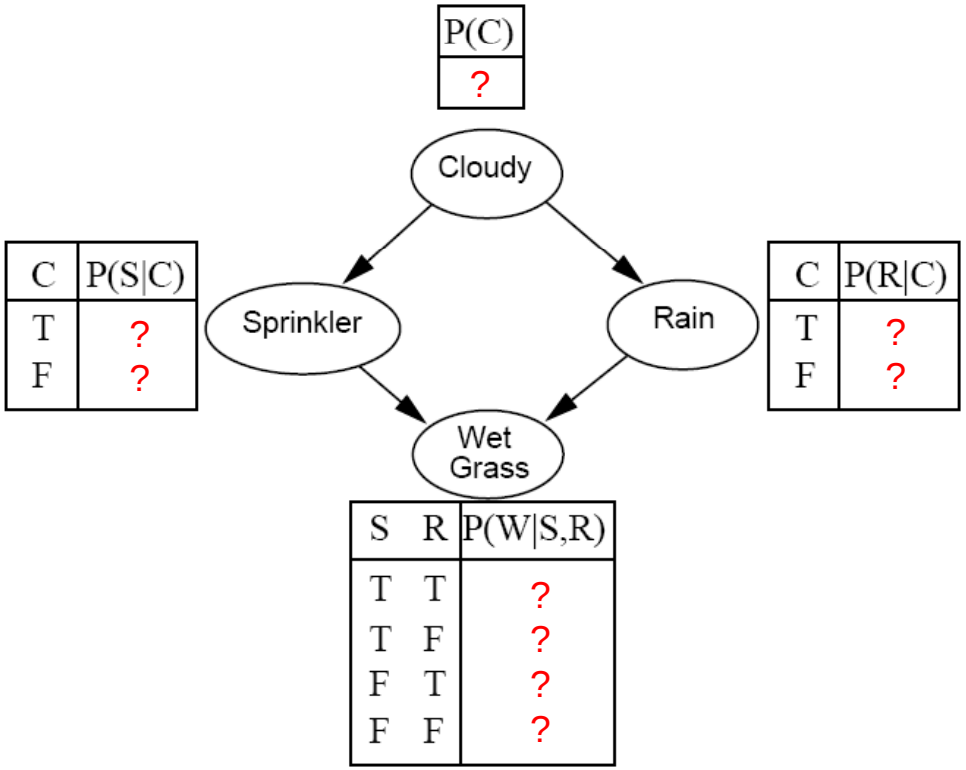
- Sample values of $u_1, \ldots, u_4$ according to $P(u_i = T) = 0.5$
  - Estimate of P(Y): # of satisfying assignments / # of sampled assignments
  - Not guaranteed to correctly figure out whether P(Y) > 0 unless you sample every possible assignment!

Other approximate inference methods

- Variational methods
  - Approximate the original network by a simpler one (e.g., a polytree) and try to minimize the divergence between the simplified and the exact model
- Belief propagation
  - Iterative message passing: each node computes some local estimate and shares it with its neighbors. On the next iteration, it uses information from its neighbors to update its estimate.

Parameter learning

- Suppose we know the network structure (but not the parameters), and have a training set of *complete* observations



Training set

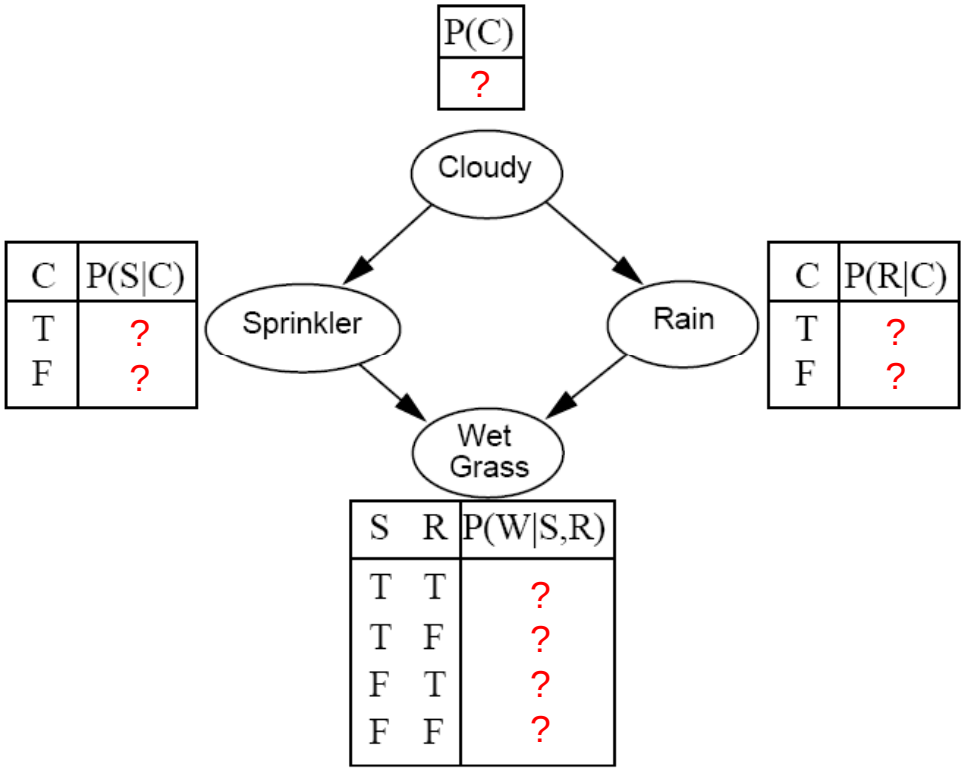
Sample	C	S	R	W
1	T	F	T	T
2	F	T	F	T
3	T	F	F	F
4	T	T	T	T
5	F	T	F	T
6	T	F	T	F
…	…	…	….	…

Parameter learning

- Suppose we know the network structure (but not the parameters), and have a training set of *complete* observations
  - P(X | Parents(X)) is given by the observed frequencies of the different values of X for each combination of parent values
  - Similar to sampling, except your samples come from the training data and not from the model (whose parameters are initially unknown)

Parameter learning

- Incomplete observations



Training set

Sample	C	S	R	W
1	?	F	T	T
2	?	T	F	T
3	?	F	F	F
4	?	T	T	T
5	?	T	F	T
6	?	F	T	F
…	…	…	….	…

Parameter learning

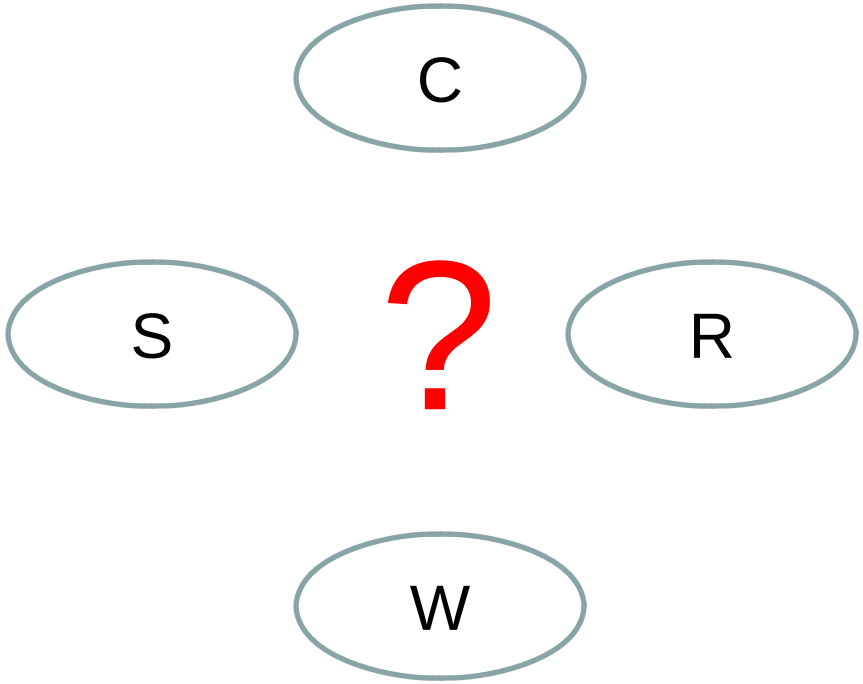
- Learning with incomplete observations:
  EM (Expectation Maximization) Algorithm

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{\boldsymbol{z}} P(\boldsymbol{Z} = \boldsymbol{z} \mid \boldsymbol{x}, \theta^{(t)}) L(\boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z} \mid \theta)$$

Z: hidden variables

Parameter learning

- What if the network structure is unknown?
  - *Structure learning* algorithms exist, but they are pretty complicated…



Training set

Sample	**C**	**S**	**R**	**W**
1	T	F	T	T
2	F	T	F	T
3	T	F	F	F
4	T	T	T	T
5	F	T	F	T
6	T	F	T	F
…	…	…	….	…