MALLA REDDY COLLEGE OF ENGINEERING

(Approved by AICTE(New Delhi), Permanently Affiliated to JNTUH)

Recognised under Section 2(f) & 12(B) of the UGC Act 1956, An ISO 9001:2015 Certified Institution.

NATIONAL BOARD OF ACCREDITATION (CSE & ECE)

Class Notes: 5th UNIT

Prepared By

Dr K Madan Mohan Ph. D.
Associate Professor,
Department of CSE (AI&ML)
MallaReddy College Of Engineering (MRCE)
Maisammaguda, Dhulapally, post via Kompally, Secunderabad - 500100

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

B.Tech. II Year II Sem.

L	T	P	C
3	0	0	3

Prerequisite: Knowledge on Data Structures.

Course Objectives:

- To learn the distinction between optimal reasoning Vs. human like reasoning.
- To understand the concepts of state space representation, exhaustive search, heuristic
- search together with the time and space complexities.
- To learn different knowledge representation techniques.
- To understand the applications of AI, namely game playing, theorem proving, and machine learning.

Course Outcomes:

- Learn the distinction between optimal reasoning Vs human like reasoning and formulate an efficient problem space for a problem expressed in natural language. Also select a search algorithm for a problem and estimate its time and space complexities.
- Apply AI techniques to solve problems of game playing, theorem proving, and machine learning.
- Learn different knowledge representation techniques.
- Understand the concepts of state space representation, exhaustive search, heuristic search together with the time and space complexities.
- Comprehend the applications of Probabilistic Reasoning and Bayesian Networks.
- Analyze Supervised Learning Vs. Learning Decision Trees

UNIT - I

Introduction to AI - Intelligent Agents, Problem-Solving Agents,
Searching for Solutions - Breadth-first search, Depth-first search, Hill-climbing search, Simulated annealing search, Local Search in Continuous Spaces.

UNIT-II

Games - Optimal Decisions in Games, Alpha–Beta Pruning, Defining Constraint Satisfaction Problems, Constraint Propagation, Backtracking Search for CSPs, Knowledge-Based Agents, **Logic-** Propositional Logic, Propositional Theorem Proving: Inference and proofs, Proof by resolution, Horn clauses and definite clauses.

UNIT-III

First-Order Logic - Syntax and Semantics of First-Order Logic, Using First Order Logic, Knowledge Engineering in First-Order Logic. Inference in First-Order Logic: Propositional vs. First-Order Inference, Unification, Forward Chaining, Backward Chaining, Resolution.
Knowledge Representation: Ontological Engineering, Categories and Objects, Events.

UNIT-IV

Planning - Definition of Classical Planning, Algorithms for Planning with State Space Search, Planning Graphs, other Classical Planning Approaches, Analysis of Planning approaches. Hierarchical Planning.

UNIT-V

Probabilistic Reasoning:
Acting under Uncertainty, Basic Probability Notation Bayes' Rule and Its Use, Probabilistic Reasoning, Representing Knowledge in an Uncertain Domain, The Semantics of Bayesian Networks, Efficient

Representation of Conditional Distributions, Approximate Inference in Bayesian Networks, Relational and First- Order Probability.

TEXT BOOK:

1. Artificial Intelligence: A Modern Approach, Third Edition, Stuart Russell and Peter Norvig, Pearson Education.

REFERENCE BOOKS:

1. Artificial Intelligence, 3rd Edn., E. Rich and K. Knight (TMH)
2. Artificial Intelligence, 3rd Edn., Patrick Henny Winston, Pearson Education.
3. Artificial Intelligence, Shivani Goel, Pearson Education.
4. Artificial Intelligence and Expert systems – Patterson, Pearson Education.

5th UNIT

Probability Reasoning

Topic-1: Acting Under Uncertainty

1. **Handling Uncertainty in Agents**

   Complexity in Belief State: If an agent sees something blurry, it must consider all possible things it could be, which makes thinking too complicated.

   Example: Seeing a shadow and considering it could be a cat, a dog, or a tree branch.

2. **Drawbacks of Contingency Plans**: Making a plan for every possible event, no matter how unlikely, makes the plan too big and impractical.

   Example: Planning for both a rainstorm and a meteorite hitting your car on the way to the airport.

3. **Uncertainty in Plan Success**: Sometimes, no plan is perfect, so the agent must choose the best option among uncertain ones.

   Example: Deciding to leave for the airport early, knowing there's no way to be 100% sure you'll be on time because of potential traffic or car issues.

4. **Making Rational Decisions Under Uncertainty**: Choose the plan that's most likely to succeed based on what the agent knows, even if it's not certain.

   Example: Leaving 90 minutes before a flight to avoid missing it, even if there's a small chance of unexpected delays.

5. **Weighing Goals and Likelihoods**: Make the best choice by considering how important each goal is and how likely it is to be achieved.

   Example: Choosing between leaving very early to definitely make the flight but waiting long at the airport, or leaving with just enough time, risking being late.

1.1 Summarizing uncertainty

Uncertain Reasoning Example: Dental Diagnosis

- Diagnosing a toothache involves uncertainty.
- Using propositional logic for diagnosis is problematic.

Problems with Logical Rules:

1. **Incomplete Rules:**
   a) Rule: `Toothache` ⇒ `Cavity` is wrong because toothaches can also be caused by gum disease, abscesses, etc.

b) **A correct rule would be:** `Toothache ⇒ Cavity ∨ GumProblem ∨ Abscess ∨ ...`, which is impractical due to the long list of possibilities.

2. **Causal Rules:**
    a) Rule: `Cavity ⇒ Toothache` is also incorrect because not all cavities cause pain.
    b) Complete and exhaustive qualifications are needed, making it impractical.

Reasons Logic Fails in Medical Diagnosis:

1. **Laziness:** Listing all exceptions and conditions is too much work.
2. **Theoretical Ignorance:** No complete theory covers all medical conditions.
3. **Practical Ignorance:** Not all tests can be run or known.

Degrees of Belief and Probability Theory:

- **Logic:** Beliefs are true, false, or unknown.
- **Probability:** Beliefs have degrees between 0 (false) and 1 (true).

Example with Probability:

- **Belief with Probability:** "There is an 80% chance (probability 0.8) that a patient with a toothache has a cavity."
    - **Derived from:** Statistical data (80% of toothache patients have cavities) or general dental knowledge.
- **Knowledge State:** Probability statements depend on the information available, not the actual state.
    a) **Example:** "The probability of a cavity given a toothache is 0.8."
    b) If additional information (e.g., gum disease history) is available: "The probability of a cavity given a toothache and gum disease history is 0.4."

Conclusion:

- Probability helps manage uncertainty by quantifying belief based on available information.
- Each probability statement is specific to a knowledge state and not contradictory.

1.2 Uncertainty and Rational Decisions:

Decision Theory and Utility

Utility Theory: It represents an agent's preferences. Each state has a utility value that reflects its usefulness to the agent. For example:

- **High utility**: White wins in chess.
- **Low utility**: Black loses in chess.

Decision Theory: Combines utility theory and probability theory to make rational choices. An agent is rational if it chooses the action with the highest expected utility. This is called the principle of Maximum Expected Utility (MEU).

Examples of Plans to the Airport

1. **A90 Plan**: Leaves 90 minutes before the flight with a 97% chance of catching the flight.
    - **Utility**: High chance of catching the flight but shorter wait at the airport.
2. **A180 Plan**: Leaves 180 minutes before the flight with a higher chance of catching the flight.
    - **Utility**: Higher chance of catching the flight but longer wait.
3. **A1440 Plan**: Leaves 24 hours before the flight.
    - **Utility**: Almost certain to catch the flight but involves an intolerable wait.

Key Concepts

- **Utility is Relative**: Depends on the agent's preferences.
    - Example: A draw in chess might be high utility for an amateur against a world champion, but low for the world champion.
- **Preferences are Subjective**: Even quirky preferences can be rational if they align with the agent's utility.
    - Example: Preferring jalapeño bubble-gum ice cream over chocolate chip is a valid preference.

Principle of Maximum Expected Utility (MEU)

- **Expected Utility**: The average utility of an action's outcomes, weighted by their probabilities.
    - **Decision**: Choose the action with the highest expected utility.

Decision-Theoretic Agent

- **Belief State**: Represents possible world states and their probabilities.
- **Action Selection**: Chooses actions based on expected utility from probabilistic predictions of outcomes.

Algorithm:

Step-1: A decision-theoretic agent that selects rational actions.

Step-2: function DT-AGENT (percept) returns an action

Step-3: persistent: belief state, probabilistic beliefs about the current state of the world

Step-4: action, the agent's action

Step-5: update belief state based on action and percept

Step-6: calculate outcome probabilities for actions,

Step-7: given action descriptions and current belief state

Step-8: select action with highest expected utility

Step-9: given probabilities of outcomes and utility information

Step-10: return action

Algorithm Explanation with Examples

a) Initialize

Explanation: Start by setting up the agent's belief state. The belief state represents the agent's understanding of the world, including uncertainties.

Example: Suppose our agent is a robot in a grid world trying to find a charging station. Initially, the robot believes the charging station could be in any of the four corners of the grid with equal probability (25% each).

b) Perceive and Update

Explanation: The agent receives a new sensory input (percept) and updates its belief state based on this new information and the last action it took.

Example: The robot moves to a new cell and detects a wall in the north direction. This new percept helps the robot update its belief state, adjusting the probabilities of where the charging station might be based on the new information.

c) Calculate Outcome Probabilities

Explanation: For each possible action, the agent calculates the probabilities of different outcomes occurring based on its current belief state and the descriptions of the actions.

Example: The robot considers moving east or south. It calculates that if it moves east, there is a 70% chance of reaching an empty cell and a 30% chance of hitting a wall. If it moves south, there is a 60% chance of reaching an empty cell and a 40% chance of hitting a wall.

d) Expected Utility Calculation

Explanation: Determine the expected utility for each action. Multiply the utility (benefit or cost) of each possible outcome by its probability and sum these to get the expected utility for each action.

Example: Assume the robot has a utility function where reaching an empty cell is +10 and hitting a wall is -10. For moving east:

- Expected utility = (0.70 * 10) + (0.30 * -10) = 7 - 3 = 4. For moving south:
- Expected utility = (0.60 * 10) + (0.40 * -10) = 6 - 4 = 2.

e) Select Action

Explanation: Choose the action that maximizes the expected utility. Compare the expected utilities from the previous step and select the action with the highest value.

Example: The robot compares the expected utilities of moving east (4) and moving south (2). Since moving east has a higher expected utility, the robot chooses to move east.

f) Execute Action

Explanation: Perform the chosen action.

Example: The robot moves east.

g) Return

Explanation: Return the action that was chosen and executed by the agent.

Example: The robot's decision to move east is returned as the chosen action.

Conclusion:

- **Initialize:** Set up initial probabilistic beliefs (e.g., charging station could be in any corner).
- **Perceive and Update:** Use sensory input to update beliefs (e.g., detect a wall).
- **Calculate Outcome Probabilities:** Assess probabilities for each action (e.g., moving east or south).
- **Expected Utility Calculation:** Compute expected utilities based on probabilities and utilities (e.g., +4 for east, +2 for south).
- **Select Action:** Pick the action with the highest expected utility (e.g., move east).
- **Execute Action:** Perform the selected action (e.g., move east).
- **Return:** Provide the chosen action (e.g., move east)

Conclusion Points:

1. **Probabilistic Reasoning**: Utilizes probabilities to manage uncertainty caused by incomplete or unreliable information.
2. **Real-World Examples**: Situations like predicting weather, assessing reactions, or predicting sports outcomes illustrate uncertainty in everyday life.
3. **AI Application**: AI uses probabilistic reasoning for unpredictable outcomes, managing numerous possibilities, and handling errors during experiments.
4. **Bayes' Rule**: Updates beliefs based on new evidence, combining prior probabilities with likelihoods.

5. **Naïve Bayes Classifier**: Applies Bayes' Rule assuming independence of features given the class, simplifying probabilistic calculations.
6. **Bayesian Networks**: Graphical models depicting variable dependencies, aiding in probabilistic inference.
7. **Markov Chain Monte Carlo (MCMC)**: Algorithm for approximating complex probability distributions in Bayesian networks.
8. **Relational Probability Models**: Extend Bayesian networks to handle relational data and complex relationships.
9. **Open Universe Probability Models**: Address uncertainties about the existence and properties of objects in varying scenarios.
10. **Handling Uncertain Knowledge**: Steps include gathering information, evaluating reliability, using probabilistic reasoning, making decisions based on expected outcomes, planning contingencies, and learning from experience.

TOPIC-1 END

2nd and 3rd TOPICS: Basic Probability Notation

1. **Basic Probability Notation**:
   - **P(A)**: Probability of event A occurring.
   - **P(A')**: Probability of event A not occurring.
   - **P(A ∩ B)**: Probability of both A and B occurring simultaneously.
   - **P(A ∪ B)**: Probability of either A or B occurring.

   Example: If the probability of rain tomorrow is P(Rain) = 0.4, then the probability of no rain would be P(No Rain) = 0.6 (since P(No Rain) = 1 - P(Rain)).

2. **Conditional Probability**:
   - **P(A | B)**: Probability of event A occurring given that event B has occurred.
   - **Bayes' Theorem**: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$.

   Example: Given that it's cloudy (B), what is the probability of rain (A)? P(Rain | Cloudy).

3. **Joint Probability**:
   - **P(A ∩ B)**: Probability of both A and B occurring.

   Example: What is the probability of both flipping two coins and getting heads on both? P(Heads on coin 1 ∩ Heads on coin 2).

4. **Expectation and Variance**:
   - **E[X]**: Expected value or mean of random variable X.
   - **Var(X)**: Variance of random variable X.

   Example: For a fair six-sided die, E[X] = 3.5 (mean value), Var(X) = 2.92 (variance).

5. **Bayesian Networks**:
   - Use graphs to represent probabilistic relationships.
   - Nodes represent random variables, edges represent dependencies.

   Example: In a medical diagnosis system, nodes could represent symptoms and diseases, and edges represent conditional dependencies based on medical knowledge.

2nd and 3rd TOPICS: Basic Probability Notation

1. **Bayes' Theorem Basics:**
   - **Definition:** Bayes' theorem calculates the probability of an event based on prior knowledge of conditions that might be related to the event.
   - **Formula:** It relates the conditional probability $P(A|B)$ to $P(B|A)$, $P(A)$, and $P(B)$.
   - **Application:** Useful in updating probabilities with new evidence.
2. **Example in Medicine (Meningitis and Stiff Neck):**
   - **Scenario:** A doctor wants to determine the probability that a patient with a stiff neck has meningitis.
   - **Given Data:**
     - Probability of a stiff neck given meningitis $P(a|b) = 0.8$.
     - Probability of meningitis in the population $P(b) = \frac{1}{30,000}$.
     - Probability of a stiff neck in the population $P(a) = 0.02$.
3. **Using Bayes' Theorem:**
   - **Objective:** Calculate $P(b|a)$, which is the probability of meningitis given a stiff neck.
   - **Steps:**
     - **Step 1:** Prior Probability $P(b) = \frac{1}{30,000}$.
     - **Step 2:** Likelihood $P(a|b) = 0.8$.
     - **Step 3:** Marginal Probability $P(a) = 0.02$.
4. **Calculation:**
   - Apply Bayes' theorem formula:

$$P(b|a) = \frac{P(a|b) \cdot P(b)}{P(a)}$$

   - Substitute the values:

$$P(b|a) = \frac{0.8 \cdot \frac{1}{30,000}}{0.02}$$

   - Simplify to find $P(b|a)$.

1. **Probability Notation**:
    - **P(A)**: Probability of event A occurring.
    - **P(A')**: Probability of event A not occurring.
    - **P(A ∩ B)**: Probability of both A and B occurring simultaneously.
    - **P(A ∪ B)**: Probability of either A or B occurring.
    - **P(A ∩ B')**: Probability of A occurring but not B.
    - **P(A' ∪ B)**: Probability of either A not occurring or B occurring.
2. **Conditional Probability**:
    - **P(A | B)**: Probability of A given that B has occurred.
    - **Bayes' Theorem**: Allows updating probabilities based on new evidence.
3. **Joint Probability**:
    - Probability of both A and B occurring simultaneously.
4. **Marginal Probability**:
    - Probability of event A occurring regardless of other events.
5. **Applications in AI**:
    - **Bayesian Networks**: Represent probabilistic relationships using DAGs.
    - **Hidden Markov Models (HMMs)**: Model systems with hidden states influencing observable events.
    - **Markov Decision Processes (MDPs)**: Model decision-making under uncertainty.
    - **Gaussian Processes (GPs)**: Used for regression and classification tasks, incorporating uncertainty in predictions.
    - **Probabilistic Graphical Models (PGMs)**: Encode conditional independence structures between random variables.
6. **Importance**:
    - **Handling Uncertainty**: Enables AI to make robust decisions despite incomplete or noisy data.
    - **Learning from Data**: Bayesian methods and probabilistic models learn from data to update beliefs.
    - **Inference**: Tools for deducing new information from existing knowledge.
    - **Decision Making**: Support for decision-making under uncertainty in various applications.

1. **Sample Space and Possible Worlds**:
    - Probabilistic assertions are about possible worlds, where each world represents a specific outcome or scenario.
    - The set of all possible worlds is called the sample space (Ω).
    - Each possible world is denoted by ω, and probabilities ($P(\omega)$) are assigned to each world, ranging from 0 to 1.
2. **Probability Model**:
    - A fully specified probability model assigns probabilities to each possible world.
    - Basic axioms ensure that probabilities range between 0 and 1, and the sum of probabilities across all possible worlds equals 1.
    - For example, when rolling fair dice, each outcome (such as (1,1), (1,2), ..., (6,6)) has an equal probability of 1/36.
3. **Events**:
    - Events in probability theory refer to sets of possible worlds that satisfy certain conditions or propositions.
    - Events are described using formal language propositions and correspond to subsets of the sample space.
    - Probability associated with an event is the sum of probabilities of all possible worlds in which the event occurs.
4. **Conditional Probability**:
    - Conditional probabilities ($P(A \mid B)$) represent the probability of event A occurring given that event B has occurred.
    - The product rule ($P(A \wedge B) = P(A \mid B) * P(B)$) describes how to compute joint probabilities from conditional probabilities.
5. **Unconditional and Conditional Probabilities**:
    - Unconditional probabilities (or priors) are probabilities without any additional information.
    - Conditional probabilities (or posteriors) are probabilities given specific evidence or conditions.
    - Conditional probabilities adjust based on observed evidence, reflecting updated beliefs or likelihoods.

6. **Decision Making and Inference**:
    - Probabilistic reasoning supports decision-making under uncertainty by quantifying degrees of belief.
    - Inference involves updating probabilities based on new evidence or observations.
    - Logical implication differs from conditional probability in that the latter accounts for additional evidence and uncertainty.

Conclusion Points:

1. **Definition and Purpose**: Bayes' Theorem, named after Thomas Bayes, calculates the conditional probability of an event given another event has occurred. It's instrumental in updating probabilities based on new information.
2. **Formula**: The theorem states: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$, where $P(A)$ and $P(B)$ are the probabilities of events A and B, $P(A|B)$ is the probability of A given B, and $P(B|A)$ is the probability of B given A.
3. **Applications**: Bayes' Theorem finds applications across various disciplines such as medicine (for diagnostic tests), finance (risk assessment), and machine learning (natural language processing and spam filtering).
4. **Components**: It involves prior probability (initial probability before new data), posterior probability (updated probability after considering new information), and conditional probability (probability of event A given event B).
5. **Mathematical Foundation**: It's derived fro the definition of conditional probability and the theorem of total probability, allowing for effective probabilistic inference.

6. Mathematical Foundation: It's derived from the definition of conditional probability and the theorem of total probability, allowing for effective probabilistic inference.

7. **Practical Use**: By incorporating prior knowledge and updating it with observed data, Bayes' Theorem facilitates more accurate predictions, enhances decision-making under uncertainty, and supports evidence-based reasoning.

8. **Historical Context**: Developed in 1763, Bayes' Theorem has become a cornerstone in statistical inference and Bayesian reasoning, influencing fields ranging from philosophy to engineering.

9. **Critical Considerations**: It requires events to have non-zero probabilities and assumes independence or conditional independence where applicable.

10. **Educational Value**: Understanding Bayes' Theorem equips individuals to analyze and interpret probabilistic relationships, fostering a deeper understanding of uncertainties in real-world scenarios.

11. **Continued Relevance**: Ongoing advancements in data analytics and machine learning underscore the enduring significance of Bayes' Theorem in extracting meaningful insights from data.

12. Bayes' Theorem provides a robust framework for updating beliefs and making informed decisions in the face of uncertain and evolving information.

Extra Information

What is Probabilistic Notation?

Probabilistic notation uses symbols and rules to talk about chances and predictions. It helps AI make decisions when things are uncertain, like guessing weather or predicting stocks.

Basic Probabilistic Notations:

1. **Probability Notation:**
   - **P(A)**: Probability that event A happens.
   - **P(A')**: Probability that event A doesn't happen.
   - **P(A ∩ B)**: Probability that both A and B happen.
   - **P(A ∪ B)**: Probability that either A or B (or both) happen.

   Example: If you toss a coin, P(Heads) = 0.5 means there's a 50% chance of getting heads.
2. **Conditional Probability:**
   - **P(A | B)**: Probability of A happening given that B has already happened.

   Example: P(Rain | Cloudy) tells us the chance of rain when it's cloudy.
3. **Joint Probability:**
   - **P(A ∩ B)**: Probability of both A and B happening together.

   Example: P(Heads ∩ Tails) = 0 because you can't get both heads and tails at the same time in one coin toss.
4. **Marginal Probability:**
   - **P(A)**: Probability of A happening, regardless of other events.

   Example: P(Sunny) = P(Sunny ∪ Cloudy) tells us the chance of sunny weather, whether it's cloudy or not.

Advanced Probabilistic Notations:

1. **Random Variables:**
   - **X**: Represents an outcome.

- **P(X = x)**: Probability that X is a specific value.

Example: X could be the number you roll on a die. P(X = 4) = 1/6 in a fair six-sided die.

2. **Probability Distributions:**
   - **PMF (Probability Mass Function)**: For discrete outcomes like dice rolls.
   - **PDF (Probability Density Function)**: For continuous outcomes like heights.

   Example: PMF tells us the chance of rolling each number on a die, while PDF shows how likely different heights are in a group of people.
3. **Expectation and Variance:**
   - **E[X]**: Expected value or average of X.
   - **Var(X)**: How spread out the possible values of X are.

   Example: For a fair die, E[X] = 3.5 (average of 1 to 6) and Var(X) = 2.92.
4. **Covariance and Correlation:**
   - **Cov(X, Y)**: How X and Y change together.
   - **Corr(X, Y)**: How strongly X and Y are related.

   Example: Cov(Height, Weight) tells us if taller people tend to weigh more, while Corr shows how strong that relationship is.

Importance of Probabilistic Notation:

- Helps AI handle uncertainty in data, like predicting weather changes.
- Lets AI learn from data and update its knowledge, like predicting which movie you might like.
- Allows AI to make smart decisions based on what it knows, even if it doesn't have all the information.

Conclusion:

- Probabilistic notation is crucial in AI because it gives a clear way for AI to understand and work with uncertain information.
- Whether predicting outcomes, learning from data, or making decisions, mastering this language helps AI become smarter and more reliable in various real-world applications.

EXTRA INFORMATION:

1. **Sample Space:**
   - **Definition:** The sample space consists of all possible outcomes of an experiment. Each outcome is a possible world.

- **Example:** Rolling two dice. The sample space Ω includes outcomes like (1,1), (1,2), ..., (6,6) totaling 36 outcomes.

2. **Probability Model:**
   - **Definition:** Assigns a probability to each possible outcome in the sample space, denoted by P(ω).
   - **Example:** If both dice are fair, each outcome in Ω has a probability of 1/36.
3. **Event:**
   - **Definition:** An event is a set of outcomes from the sample space.
   - **Example:** "The total is 11" is an event. It includes outcomes (5,6) and (6,5). P(Total = 11) = P((5,6)) + P((6,5)) = 1/36 + 1/36 = 1/18.
4. **Unconditional (Prior) Probability:**
   - **Definition:** Probability assigned to an event without any additional information.
   - **Example:** P(doubles) when rolling fair dice is 1/6.
5. **Conditional (Posterior) Probability:**
   - **Definition:** Probability of an event given that another event has occurred.
   - **Example:** P(doubles | first die is 5) is the probability of rolling doubles given that the first die shows 5. It adjusts the probability based on new information.
6. **Product Rule:**
   - **Definition:** Relates conditional probability to unconditional probability.
   - **Example:** P(A and B) = P(A | B) * P(B). For example, P(doubles and first die is 5) = P(doubles | first die is 5) * P(first die is 5).
   - Probability theory helps us quantify uncertainty and make decisions based on observed evidence. It provides a formal framework to understand how likely events are and how they relate to each other in terms of probability.

2nd and 3rd TOPICS are END

4th Topic: Probabilistic Reasoning

I. Probabilistic Reasoning:

Probabilistic reasoning is a method used in intelligent systems to handle uncertainty. It uses probabilistic concepts to make decisions or draw conclusions based on available information.

Problems of Full Joint Probability Distribution:

1. **Complexity with Many Variables**: When the number of variables increases, the full joint probability distribution becomes extremely large and difficult to manage.

   Example: Imagine trying to calculate the probabilities of all possible outcomes in a complex scenario involving many factors, like predicting weather patterns based on numerous environmental variables.

2. **Tedious Specification**: Specifying probabilities for every possible scenario sequentially can be laborious and prone to errors.

   Example: Assigning probabilities for every combination of symptoms and diseases in a medical diagnosis system without any structured framework.

Bayesian Networks:

Bayesian networks, also known as belief networks or causal networks, are a structured way to represent and reason about probabilistic relationships. They address the limitations of full joint probability distributions.

Properties of Bayesian Networks:

a) **Node Representation**: Each node in a Bayesian network represents a random variable, which can be discrete (like true/false) or continuous (like temperature).

Example*:* In a medical diagnosis system, nodes could represent variables such as symptoms (fever, cough) or diseases (flu, pneumonia).

b) **Directed Graph Structure**: Nodes are connected by directed arrows that show the influence between variables. An arrow from node X to Y means X directly influences Y.

Example: An arrow from "Smoking" to "Lung Cancer" in a health-related Bayesian network signifies that smoking affects the likelihood of developing lung cancer.

c) **Directed Acyclic Graph (DAG)**: It's a graph without any cycles (loops), ensuring there are no feedback loops where a variable indirectly affects itself.

Example: A DAG ensures that in a system predicting financial market trends, no variable (like stock price) influences itself through a series of dependent variables.

d) **Conditional Probability Distribution**: Each node has associated probabilities that depend on its parents in the network. This distribution captures how each variable is affected by its direct influences.

Example: For "Rain" as a node, its conditional probability distribution might specify how likely rain is given the previous day's weather and current atmospheric conditions.

Bayesian networks provide a structured approach to probabilistic reasoning by organizing variables, their relationships, and their uncertainties, making complex systems more manageable and allowing for efficient decision-making and inference.

Extra Information:

1. **Definition and Importance of Probabilistic Reasoning**:
   - Probabilistic reasoning in AI uses probability theory to manage uncertainty in decision-making.
   - It enables AI systems to function effectively in complex, real-world scenarios where information is incomplete or noisy.
2. **Key Techniques**:
   - **Bayesian Networks**: Graphical models showing relationships between variables and their probabilities, used in medical diagnosis and causal reasoning.
   - **Markov Models**: Predict future states based on current and past states, widely used in weather forecasting and speech recognition.
   - **Hidden Markov Models (HMMs)**: Extend Markov models to include hidden states, crucial in applications like stock market prediction.
   - **Probabilistic Graphical Models**: Framework encompassing Bayesian networks and HMMs, suitable for complex relationships.
3. **Applications**:
   - **Robotics**: Enables robots to navigate uncertain environments (e.g., SLAM algorithms).
   - **Healthcare**: Aids in medical diagnosis by assessing disease likelihood from symptoms.
   - **Natural Language Processing (NLP)**: Used for tasks like part-of-speech tagging and machine translation.
   - **Finance**: Models market behavior and assesses risks in investment decisions.
4. **Advantages**:
   - **Flexibility**: Adaptable to various domains and types of uncertainty.
   - **Robustness**: Handles noise and incomplete data well, making it reliable in practical applications.

- **Interpretable**: Provides a clear framework for understanding uncertainty, aiding in transparency and decision-making.

5. **Role in AI Systems**:
   - Quantifies uncertainty by assigning probabilities rather than simple true/false outcomes.
   - Uses evidence and past experience to refine probabilities and make informed decisions.
   - Enables effective decision-making in uncertain environments, improving reliability and performance of AI systems.
6. **Conclusion**:
   - Probabilistic reasoning is essential for creating intelligent AI systems that operate effectively in real-world conditions.
   - It enhances decision-making by acknowledging and quantifying uncertainty, ultimately improving reliability and performance.
   - These points highlight how probabilistic reasoning is fundamental in advancing AI capabilities, enabling them to handle the complexities and uncertainties inherent in real-world applications.

Extra Information:

1. **Definition of Probabilistic Reasoning:** Probabilistic reasoning integrates probability theory with logic to manage uncertainty in knowledge representation. It allows us to quantify the likelihood of different outcomes or states of the world based on available evidence or assumptions.
2. **Uncertainty in Knowledge Representation:** Traditional approaches like propositional and first-order logic assume certainty in predicates (statements about the world). In contrast, probabilistic reasoning acknowledges uncertainty, where we are unsure about the truth value of predicates due to various factors like unreliable sources, experimental errors, or environmental variations.
3. **Causes of Uncertainty:** Several factors contribute to uncertainty in real-world scenarios, including unreliable information sources, experimental errors, equipment faults, temperature variations, and broader phenomena like climate change. These factors challenge the certainty of statements we might make about the world.
4. **Application Scenarios:** Probabilistic reasoning is essential in AI when dealing with:
   a) **Unpredictable Outcomes:** Situations where outcomes cannot be determined with certainty, such as weather predictions or human behavior modeling.

b) **Complex Specifications:** When the number of possible states or outcomes becomes too vast to handle with certainty using traditional logic.

c) **Unknown Errors:** Cases where unexpected errors or anomalies occur during experiments or data collection, affecting the reliability of information.

5. **Role of Probability:** Probability provides a structured framework to express and manage uncertainty in probabilistic reasoning. It allows AI systems to make informed decisions or predictions by quantifying uncertainty and updating beliefs based on new evidence.
6. **Practical Examples:** Examples of probabilistic reasoning applications include predicting the likelihood of rain, modeling the behavior of individuals in various contexts, or assessing the outcome of competitive events like sports matches.
7. Probabilistic reasoning is crucial in AI for handling uncertainty effectively, which is pervasive in real-world scenarios due to various factors and phenomena beyond our complete control or knowledge. It enriches knowledge representation by allowing AI systems to reason about probabilities and make decisions in the face of uncertainty.

Conclusion Points

1. **Handling Uncertain Knowledge**: Probabilistic reasoning deals with uncertain information by assigning probabilities to events, quantifying the likelihood of their occurrence.
2. **Representation of Probabilities**: Probabilities are represented using prior probability distributions, which express beliefs about the likelihood of events before observing any evidence.
3. **Bayes' Rule**: This fundamental rule in probability theory allows for updating beliefs (posterior probability) based on new evidence, combining prior beliefs and likelihoods.
4. **Naïve Bayes Classifier**: A simple probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between features.
5. **Bayesian Network**: A graphical representation of probabilistic relationships among variables, using nodes (variables) and edges (dependencies), providing a compact way to model joint probability distributions.
6. **Inference in Bayesian Networks**: Techniques include exact methods (e.g., variable elimination) and approximate methods (e.g., Markov Chain Monte Carlo) to compute probabilities over variables.
7. **Relational Probability Models**: Extend Bayesian networks to handle relationships and dependencies between entities in relational databases or knowledge bases.

8. **Markov Chain Monte Carlo (MCMC)**: An algorithm used for sampling from probability distributions, particularly useful in Bayesian inference and approximate inference in Bayesian networks.
9. **First-Order Models**: Challenges arise in applying probabilistic reasoning to complex domains where relationships and rules involve first-order logic, requiring specialized approaches.
10. **Open Universe Probability Models**: Address the issue of unknown or changing sets of entities in probabilistic reasoning, allowing for more flexible and adaptive models.
11. These points highlight how probabilistic reasoning and Bayesian methods are applied in AI systems to handle uncertainty and make informed decisions based on probabilistic inference.

4th Topic END

5th Topic: Representing Knowledge in an Uncertain Domain:

a) How Knowledge is Represented

Knowledge Representation is how we store information about the world so that a computer can understand and use it to make decisions.

- **Example**: Think of it like writing down a recipe for cooking. The recipe tells you the ingredients and steps needed to make a dish. In the same way, knowledge representation tells a computer the important facts and rules about a situation.

b) Knowledge Representation Techniques

There are several techniques for representing knowledge, especially in uncertain domains where information may not be complete or certain:

1. **Logic-Based Representation**:
    - **Propositional Logic**: Simple statements that are either true or false.
        - ***Example*:** "It is raining" (true or false).
    - **First-Order Logic**: Uses objects, relations, and quantifiers.
        - ***Example*:** "All cats are mammals."
2. **Probabilistic Representation**:
    - **Bayesian Networks**: Graphical models that represent probabilistic relationships among variables.
        - ***Example*:** Representing the likelihood of having a cold given symptoms like cough and fever.
3. **Fuzzy Logic**:
    - Deals with reasoning that is approximate rather than fixed and exact.
        - ***Example*:** "The room is warm" rather than a specific temperature.
4. **Semantic Networks**:
    - Graph structures for representing knowledge in patterns of interconnected nodes and edges.
        - ***Example*:** Representing relationships like "A dog is a pet" and "A pet can be a companion."

c) Role of Knowledge in Reasoning

Reasoning is the process of drawing conclusions from available knowledge.

- **Example**: If you know that "All humans are mortal" and "Socrates is a human," you can reason that "Socrates is mortal."

Knowledge helps in reasoning by providing the facts and rules needed to make logical deductions, predictions, or decisions. In uncertain domains, probabilistic reasoning helps in making decisions based on incomplete or uncertain information.

- **Example**: Given symptoms like fever and cough, reasoning with a Bayesian network can help diagnose the probability of having a flu.

d) Issues in Knowledge Representation

There are several challenges when it comes to representing knowledge, especially in uncertain domains:

1. **Complexity**:
    - Representing real-world situations can be very complex.
        - ***Example*:** Accurately modeling the weather involves many variables and relationships.
2. **Uncertainty**:
    - Not all information is certain; there can be unknowns or probabilities.
        - ***Example*:** Predicting if it will rain tomorrow based on current weather data.
3. **Ambiguity**:
    - The same information can be interpreted in different ways.
        - ***Example*:** The word "bank" can mean the side of a river or a financial institution.
4. **Incompleteness**:
    - Sometimes, not all information is available.
        - ***Example*:** Diagnosing a disease without knowing all symptoms.
5. **Scalability**:
    - Handling large amounts of knowledge efficiently.
        - ***Example*:** A search engine indexing billions of web pages.

- Knowledge representation in uncertain domains involves storing information in ways that allow computers to understand and use it for reasoning.
- Techniques like logic-based representation, probabilistic models, and fuzzy logic help manage uncertainty. Knowledge plays a critical role in reasoning by providing the necessary facts and rules.
- However, issues like complexity, uncertainty, ambiguity, incompleteness, and scalability pose challenges to effective knowledge representation.

5th Topic END

Topic-6: The Semantic of Bayesian Networks:

Semantics: The meaning or interpretation of a concept or term within a specific context.

Example: In a Bayesian Network, the semantics refer to the specific joint probability distribution over all the variables represented by the network.

What is a Bayesian Network?

A Bayesian Network is like a map showing how different factors are related to each other and how likely certain things are to happen. It has two main parts:

1. **Nodes**: These represent different factors or events (like whether it's raining, whether the grass is wet, etc.).
2. **Links**: These show how the factors are connected (like if it's raining, then there's a chance the grass is wet).

How Does It Work?

1. **Nodes and Conditional Probabilities**: Each node has some numbers attached to it called conditional probabilities. These numbers tell us how likely an event is, given some other events. For example, if we have a node for "Rain" and a node for "Wet Grass", the conditional probability tells us how likely the grass is wet if it's raining.
2. **Conditional Probability Tables (CPTs)**: Each node has a table (CPT) that shows the conditional probabilities. For example, the "Wet Grass" node will have a table showing the probability of the grass being wet if it's raining and if it's not raining.

Example

Imagine we have three nodes: Rain, Sprinkler, and Wet Grass.

- **Rain**: It can either rain or not rain.
- **Sprinkler**: It can either be on or off.
- **Wet Grass**: The grass can either be wet or dry.

Conditional Probabilities:

- If it rains, there's an 80% chance the grass will be wet.
- If the sprinkler is on, there's a 90% chance the grass will be wet.
- If it's raining and the sprinkler is on, the grass is almost certainly wet.

Full Joint Distribution

The full joint distribution is like a giant table that lists all possible combinations of events (rain, no rain, sprinkler on, sprinkler off) and their probabilities. This table helps us answer any question about the domain (like what's the probability the grass is wet given that it's raining and the sprinkler is off).

Chain Rule

The chain rule lets us calculate the probability of any combination of events by multiplying the conditional probabilities together.

For example, to find the probability of it raining and the grass being wet, we multiply the probability of it raining by the probability of the grass being wet given that it's raining.

Why Bayesian Networks Are Useful

- **No Redundancy**: Bayesian Networks don't repeat any probability values, so there's no chance of conflicting information.
- **Accurate Representation**: It's impossible to create a Bayesian Network that doesn't follow the rules of probability, ensuring accuracy.

In Simple Terms

- **Nodes**: Think of them as questions (Is it raining? Is the sprinkler on? Is the grass wet?).
- **Links**: These are the connections showing how one question affects another (Rain affects Wet Grass).
- **CPTs**: These are small tables that tell us the answer to one question based on the answers to others.
- **Chain Rule**: This is like a recipe for calculating the probability of any combination of answers.

With a Bayesian Network, you can figure out the likelihood of any event based on the known relationships and probabilities, and it's always consistent with the rules of probability.

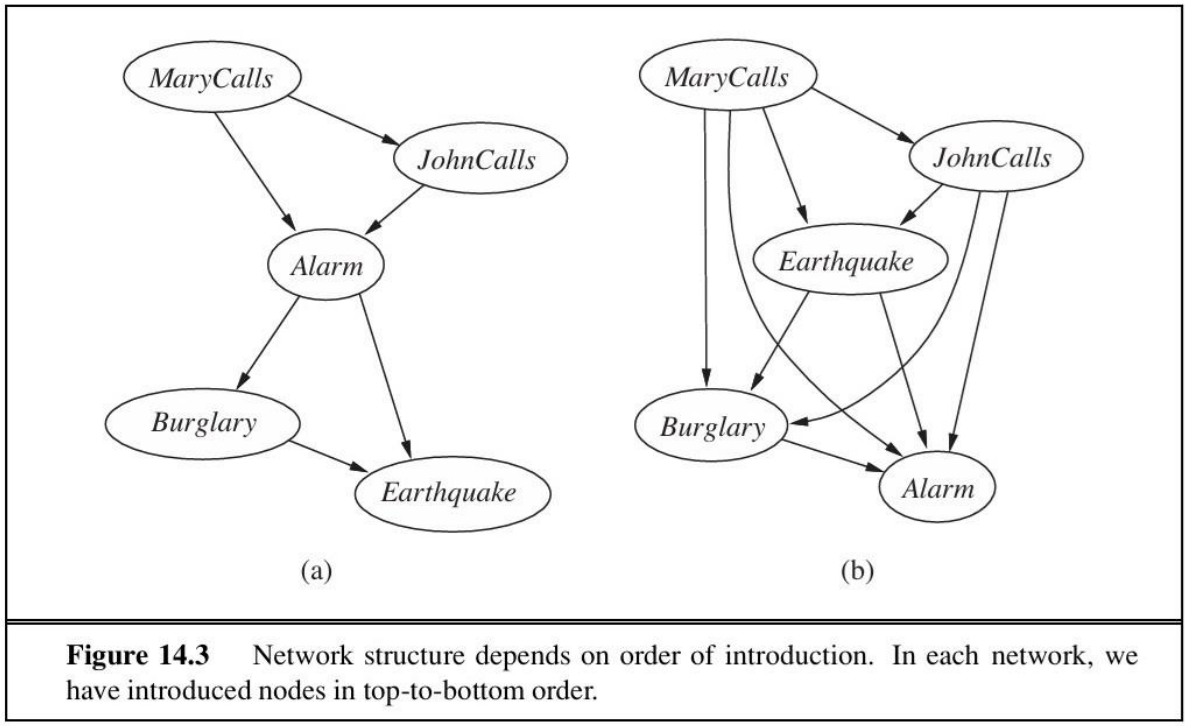

Figure 14.3 Network structure depends on order of introduction. In each network, we have introduced nodes in top-to-bottom order.

Compactness of Bayesian Networks

Example: Burglar Alarm Network

Imagine a network where:

- "Burglary" and "Earthquake" can cause an "Alarm."
- If the "Alarm" goes off, "John" and "Mary" might call the police.

Why Bayesian Networks are Compact:

1. **Local Structure:**
   - Each variable (e.g., "Alarm") depends directly on only a few other variables (e.g., "Burglary" and "Earthquake"), not all variables in the network.
   - This local dependency keeps the network manageable.
2. **Conditional Probabilities:**
   - We only need probabilities for direct influences (e.g., the probability that the alarm goes off given a burglary and/or an earthquake), not for every possible combination of all variables.

Numbers Example:

- Suppose we have 30 variables (like "Alarm," "JohnCalls," etc.), each influenced by at most 5 others.
- For each variable, we need at most 252^525 (or 32) numbers to specify its probabilities.
- Total numbers needed: 30×32=96030 \times 32 = 96030×32=960.
- If we specified the full joint distribution, we'd need 2302^{30}230 (over a billion) numbers.

Practical Considerations:

1. **Simplicity vs. Accuracy:**

   Adding more connections (like linking "Earthquake" directly to "JohnCalls" and "MaryCalls") might improve accuracy but increases complexity.

   Example Decision:

   - **Without the link:** John and Mary call based on the alarm.
   - **With the link:** They might not call during an earthquake because they assume it's the cause of the alarm.
   - Whether to add this link depends on if the small gain in accuracy is worth the added complexity.

By using Bayesian networks, we can efficiently manage and compute probabilities in large domains by focusing on local structures and direct dependencies.

If we choose the wrong order when creating a Bayesian network, we can end up with a more complex network that requires more probabilities to be specified and includes difficult and unnatural relationships.

Example: Burglary Scenario

Consider a scenario involving an alarm system that detects both burglaries and earthquakes, and two people (John and Mary) who call when they hear the alarm.

Correct Node Order: Burglary, Earthquake, Alarm, JohnCalls, MaryCalls

In this correct order:

1. **Burglary** and **Earthquake** are independent events.
2. **Alarm** depends on both Burglary and Earthquake.
3. **JohnCalls** and **MaryCalls** depend on the Alarm.

This order results in a simple and compact network.

Incorrect Node Order: MaryCalls, JohnCalls, Alarm, Burglary, Earthquake

In this incorrect order:

1. **MaryCalls** has no parents.
2. **JohnCalls** depends on MaryCalls.
3. **Alarm** depends on both MaryCalls and JohnCalls.
4. **Burglary** depends on the Alarm.
5. **Earthquake** depends on both Alarm and Burglary.

This creates a more complex network with unnecessary relationships, making it harder to specify and understand the probabilities. For example, you would need to assess the probability of an Earthquake given both an Alarm and a Burglary, which is unnatural.

Very Bad Node Order: MaryCalls, JohnCalls, Earthquake, Burglary, Alarm

In this very bad order:

1. **MaryCalls** and **JohnCalls** have no parents.
2. **Earthquake** depends on both JohnCalls and MaryCalls.
3. **Burglary** depends on Earthquake.
4. **Alarm** depends on both Burglary and Earthquake.

This results in a network as complex as specifying the full joint distribution, requiring 31 distinct probabilities.

Key Point

Using a causal model (causes to effects) is simpler and requires fewer and more natural probabilities than using a diagnostic model (symptoms to causes). For example, in medicine,

doctors prefer giving probabilities for causes leading to symptoms rather than the other way around.

Conclusion:

- Correct order = simpler network, fewer probabilities.
- Incorrect order = more complex network, more probabilities, unnatural relationships.
- Very bad order = maximum complexity, as complex as the full joint distribution.

Conditional independence in Bayesian networks using simple examples:

1. **Numerical Semantics:** This means defining how variables interact through specific probabilities in the network.

   Example: In a Bayesian network about home security, if Alarm going off influences whether John Calls and Mary Calls, we might say:

   - P(John Calls | Alarm) = 0.8 (John is likely to call if the alarm goes off).
   - P(Mary Calls | Alarm) = 0.6 (Mary is somewhat likely to call if the alarm goes off).

2. **Topological Semantics:**

   This focuses on the structure of the network to determine independence relationships.

   Example: In the same security network, if Alarm directly affects both John Calls and Mary Calls, they would be independent of each other given Alarm.

   If Alarm is on, whether John calls or not doesn't change the probability of Mary calling, and vice versa.

3. **Equivalence of Semantics:**

   Both numerical and topological views are equivalent—they describe the same relationships.

   Example: Knowing that in our network, Burglary is independent of John Calls and Mary Calls given Alarm and Earthquake allows us to precisely calculate how these variables influence each other numerically.

4. **Markov Blanket:**

   It includes a node's parents, children, and children's parents, making it a shield against influences from other parts of the network.

   Example: If Burglary's Markov blanket includes Alarm and Earthquake, then Burglary is independent of other variables like John Calls and Mary Calls given these factors.

- Bayesian networks use both graphical (topological) and probability-based (numerical) approaches to show how variables interact.
- Understanding these concepts helps in modeling dependencies effectively, ensuring that we can make accurate predictions and decisions based on the network's structure and parameters.

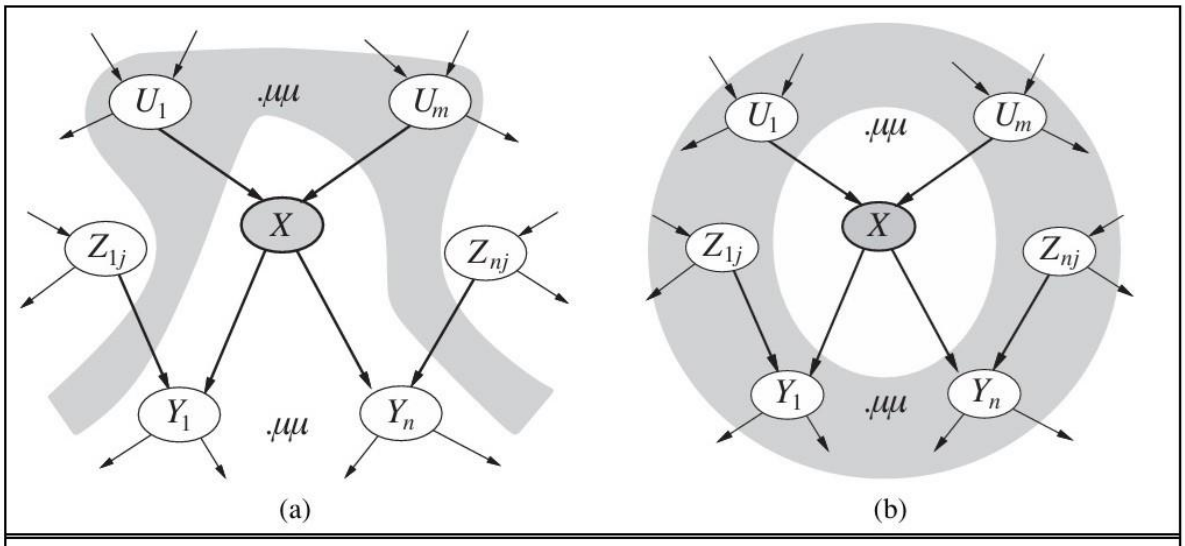

Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

(a) Conditional Independence Given Parents:

In a family tree, if X is a child and U are its parents, X is independent of its non-descendant relatives (Z) given its parents (U). Knowing X's parents (like John and Mary) tells you everything about X that knowing their cousins wouldn't.

(b) Conditional Independence Given Markov Blanket:

In a network, a node X is independent of all other nodes if you know everything about its Markov blanket. This blanket includes X's parents, children, and their other parents. For example, knowing a student's friends, their parents, and the student's own parents is enough to predict the student's behavior without needing details about other students or their families.

End of 6th Topic

Topic-7: Efficient Representation Of Conditional Distributions

Canonical Distribution:

Creating Conditional Probability Tables (CPTs) for Bayesian networks can be complex, especially with fewer parent nodes (k). In the worst case, filling these tables could need O(2^k) entries. However, many parent-child relationships follow standard patterns, described by simpler distributions. For instance, in a network predicting rain (R) with Cloudy (C) and Windy (W) as parents, instead of listing all combos (4), you might note rain is likelier when cloudy: P(R | C, W) = High if C = True, else Low.

This simplifies the CPT by using patterns rather than every scenario, making it easier to manage.

Deterministic Nodes:

In a Bayesian network, a deterministic node's value is exact and depends solely on its parent nodes, without uncertainty:

- **Logical Example:** A child node "North American" with parent nodes "Canadian," "US," and "Mexican" simply reflects whether any parent is true.
- **Numerical Example 1:** If parent nodes are car prices at different dealers, the child node "bargain hunter's price" would be the minimum price among the parents.
- **Numerical Example 2:** For a lake's parent nodes representing inflows and outflows, the child node "change in water level" would be the difference between total inflows and outflows.

Noisy-OR

Noisy logical relationships, like noisy-OR, handle uncertainty in connections. For example, in propositional logic, we might say having a Fever could be due to Cold, Flu, or Malaria. But unlike a strict OR, where having any one of these conditions guarantees a Fever, noisy-OR allows for cases where having Cold alone might not result in a Fever.

Noisy-OR Assumption

1. **Assumption 1: Listing Possible Causes**:
   When identifying causes of a symptom (like fever), we assume we've listed all major causes. If some causes are unknown or miscellaneous, we can add a "leak node" to cover these.

2. **Assumption 2: Independent Inhibition**:

   Each cause's ability to prevent the symptom (fever) is independent of others. For instance, what stops Malaria from causing a fever is separate from what stops Flu.

3. **Probability Calculation**:

   We assign probabilities (inhibition probabilities) to each cause preventing the symptom:

   - For Cold (q_{cold}): 0.6
   - For Flu (q_{flu}): 0.2
   - For Malaria ($q_{malaria}$): 0.1

4. **Building Conditional Probability Table (CPT)**:

   Using the noisy-OR model:

   - If a cause is present, it can contribute to the symptom (fever).
   - If multiple causes are present, their combined effect is considered.

5. **Example CPT Calculation for Fever**

Cold	Flu	Malaria	P(Fever)	P(¬Fever)
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	0.02
T	F	F	0.4	0.6
T	F	T	0.94	0.06
T	T	F	0.88	0.12
T	T	T	0.988	0.012

   - These values show the probability of fever (P(Fever)) or no fever (P(¬Fever)) given combinations of causes (Cold, Flu, Malaria).

6. **Advantages of Noisy-OR Model**

   - Simplifies complex probability assessments by focusing on how each cause independently influences the symptom.
   - Reduces the number of parameters needed compared to a full probability table, making it easier to manage and learn from data.

- This approach, used in medical diagnosis and other fields, helps model probabilistic relationships efficiently while accounting for uncertainty and independence among causes.

Bayesian nets with continuous variables

DISCRETIZATION

- Handling continuous variables in Bayesian networks is tricky because they can take on countless values. To handle this, we often use discretization. For instance, instead of considering every possible temperature, we might group them into categories like "cold," "warm," and "hot."
- Discretization simplifies things by reducing the number of specific values we deal with. However, it can make our estimates less precise and require larger tables of probabilities.
- Alternatively, we can use standard math functions like the Gaussian (normal) distribution, which uses parameters like mean (μ) and variance (σ^2). This approach helps us model values more accurately without discretizing them explicitly.
- In more complex situations, we might use nonparametric methods. Here, distributions are represented indirectly through specific examples with known values of related variables.
- These methods make Bayesian networks useful for real-world problems involving continuous data, balancing accuracy with practicality in calculations.

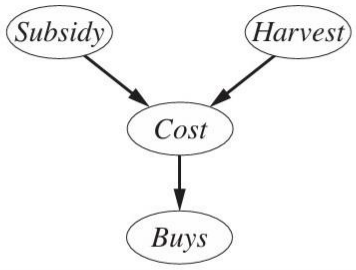

Figure 14.5 A simple network with discrete variables (*Subsidy* and *Buys*) and continuous variables (*Harvest* and *Cost*).

Hybrid Bayesian Network

In a hybrid Bayesian network, which deals with both discrete and continuous variables, we specify how variables depend on each other using conditional distributions:

1. **Continuous Variable (Cost)**:
    - **Given Parents (Harvest and Subsidy)**: We need to define how the cost (a continuous variable) depends on its parents (which can be both continuous and discrete).

- **Handling Discrete Parents**: For discrete parents like Subsidy, we specify different scenarios (Subsidy is true or false) by providing separate distributions (P(Cost | Harvest, Subsidy) and P(Cost | Harvest, ¬Subsidy)).
- **Handling Continuous Parent (Harvest)**: We use a linear Gaussian distribution, where the cost's distribution (like its mean and variability) changes linearly with the value of Harvest.
- **Example**: If Harvest affects Cost, we might say Cost follows a Gaussian distribution with mean μ = ath + bt and standard deviation σ. Here, ath and bt are parameters that define how Harvest influences Cost.

2. **Discrete Variable (Buys)**:
   - **Given Continuous Parent (Cost)**: For Buys (a discrete variable), its distribution depends on the continuous variable Cost.
   - **Example**: Buys could be influenced by Cost such that the probability of buying fruit might increase as Cost decreases. This relationship is specified in the Bayesian network to show how Cost affects Buys.

In a hybrid Bayesian network:

- Continuous variables like Cost are described using linear Gaussian distributions, with parameters that show how they depend on continuous and discrete parent variables.
- Discrete variables like Buys are described by how they depend on continuous parent variables, showing the likelihood of different outcomes based on the value of the continuous parent.

This structure helps model complex relationships in systems where both types of variables interact.

$$P(c \mid h, subsidy) = N(a_t h + b_t, \sigma_t^2)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_t h+b_t)}{\sigma_t}\right)^2}$$

$$P(c \mid h, \neg subsidy) = N(a_f h + b_f, \sigma_f^2)(c) = \frac{1}{\sigma_f \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{c-(a_f h+b_f)}{\sigma_f}\right)^2}.$$

For this example, then, the conditional distribution for Cost is specified by naming the linear Gaussian distribution and providing the parameters at, bt, σt, af, bf,and σf.

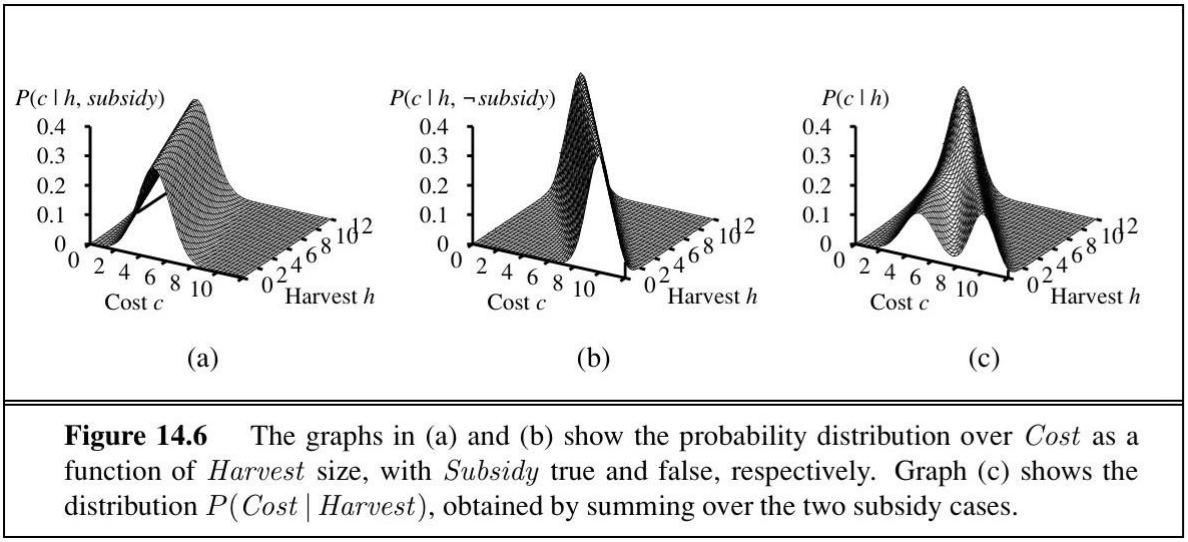

Figure 14.6 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(Cost \mid Harvest)$, obtained by summing over the two subsidy cases.

Figures 14.6(a) and (b) show these two relationships. Notice that in each case the slope is negative, because cost decreases as supply increases. (Of course, the assumption of linearity implies that the cost becomes negative at some point; the linear model is reasonable only if the harvest size is limited to a narrow range.)

Figure 14.6(c) shows the distribution P(c|h), averaging over the two possible values of Subsidy and assuming that each has prior probability 0.5. This shows that even with very simple models, quite interesting distributions can be represented.

CONDITIONAL GAUSSIAN

In networks with linear Gaussian distributions:

- All continuous variables connected in a certain way form a joint distribution that looks like a bell curve, known as a multivariate Gaussian.
- When discrete variables influence continuous ones, the conditional distribution of the continuous variables given specific values of the discrete ones is called a conditional Gaussian distribution.

For example, imagine predicting if a customer buys a product based on its cost. Let's say cost is continuous, and the buying decision (Buys) is discrete (yes or no). If we think customers are more likely to buy when the cost is low and less likely as it increases, we can model this with a soft threshold.

To create soft thresholds, we use the cumulative standard normal distribution function, Φ(x).

For instance, the probability a customer buys given a cost (c) might be $\Phi((-c+\mu)/\sigma)$, where:

$$P(\text{buys} \mid \text{Cost} = c) = \Phi((-c+\mu)/\sigma)$$

- μ is the cost where the probability of buying is 50%,
- σ controls how quickly the probability changes around μ.

As cost moves away from μ, the probability of buying decreases smoothly according to the standard normal curve. This method helps us model complex decision-making in a clear, probabilistic way.

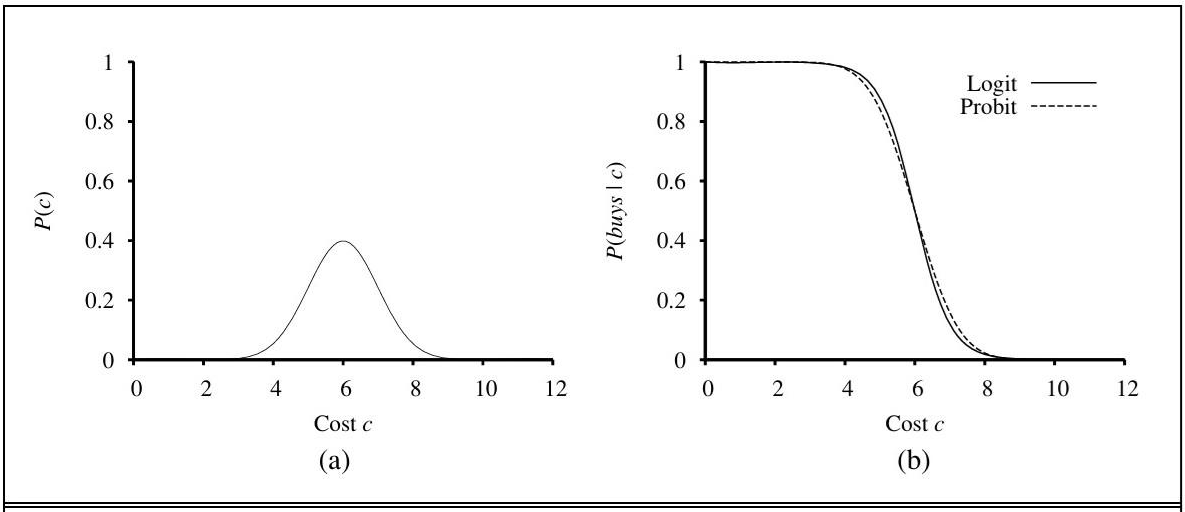

Figure 14.7 (a) A normal (Gaussian) distribution for the cost threshold, centered on $\mu = 6.0$ with standard deviation $\sigma = 1.0$. (b) Logit and probit distributions for the probability of *buys* given *cost*, for the parameters $\mu = 6.0$ and $\sigma = 1.0$.

- **This probit distri**bution (pronounced "pro-bit" and short for "probability unit") is illustrated in Figure 14.7(a). The form can be justified by proposing that the underlying decision process has a hard threshold, but that the precise location of the threshold is subject to random Gaussian noise.
- An alternative to the probit model is the logit distribution (pronounced "low-jit"). It uses the logistic function $1/(1 + e-x)$ to produce a soft threshold: $P(\text{buys} \mid \text{Cost} = c) = 1/\ 1+\exp(-2^{-c+\mu/\sigma})$.
- This is illustrated in Figure 14.7(b). The two distributions look similar, but the logit actually has much longer "tails." The probit is often a better fit to real situations, but the logit is sometimes easier to deal with mathematically. It is used widely in neural networks. Both probit and logit can be generalized to handle multiple continuous parents by taking a linear combination of the parent values.

End of Topic -7

8th Topic: APPROXIMATE INFERENCE BAYESIAN NETWORKS

MONTECARLO

- Big and complex networks make exact answers hard to find. So, we turn to approximate methods like Monte Carlo algorithms, such as simulated annealing. These methods randomly sample data to give results close to the right ones. They're widely used in science to estimate tricky values.
- For example, imagine predicting an election outcome in a large country with many factors affecting voters. Monte Carlo methods help us estimate different outcomes by randomly sampling voter opinions and behaviors.
- When calculating posterior probabilities, we look at two main types of algorithms: direct sampling and Markov chain sampling. Each tackles complex network data differently to give useful estimates.
- Other methods like variational approaches and loopy propagation are also mentioned as ways to handle similar problems at the end of the chapter.

Direct sampling methods

1. **Direct Sampling Overview:** Sampling methods generate random outcomes from known probability distributions. For example, flipping an unbiased coin with a 50% chance for heads and tails can be simulated by generating a random number between 0 and 1. If it's $\leq$ 0.5, it's heads; otherwise, it's tails.
2. **Simple Example:** To simulate a coin flip, generate a random number between 0 and 1. $\leq$ 0.5 = heads, > 0.5 = tails.
3. **Sampling in Bayesian Networks:** Bayesian networks depict probabilistic relationships among variables. Sampling follows a topological order, starting with variables without parents and using their distributions to sample values. Proceed to variables with known parent values using conditional distributions until all are sampled.
4. **Conclusion** In a Bayesian network where variables like Weather, Temperature, and Outfit Choice are interconnected, sample in topological order: Weather first, then Temperature, and finally Outfit Choice, respecting each variable's dependencies. This approach ensures that sampled values reflect the network's probabilistic structure accurately.

function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

$\mathbf{x} \leftarrow$ an event with n elements
foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
  $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
return x

Figure 14.13 A sampling algorithm that generates events from a Bayesian network. Each variable is sampled according to the conditional distribution given the values already sampled for the variable's parents.

Algorithm Explanation:

The function PRIOR-SAMPLE that generates events from a Bayesian network

1. Start with an empty event x containing n elements.
2. For each variable X_i in the Bayesian network:
   - Sample a value $x[i]$ for X_i based on its conditional distribution given the values already sampled for its parent variables.
3. Return the event x, which represents a sample from the prior distribution specified by the Bayesian network bn.

PRIOR-SAMPLE constructs a sample by iteratively sampling each variable in the network according to its conditional probability distribution, given the values already sampled for its parent variables.

1. **Sampling Process**: We use a method called PRIOR-SAMPLE to generate events from a network (like a Bayesian network). Each event depends on its parent events.
2. **Probability Calculation**: The probability of generating a specific event using PRIOR-SAMPLE is the product of the probabilities of its components given their parents.
3. **Accuracy of Sampling**: As we increase the number of samples (N), the fraction of times a specific event occurs (NPS(x1,...,xn)/N) gets closer to its actual probability (SPS(x1,...,xn)).
4. **Estimating Probabilities**: If we generate a large number of samples, we can estimate the probability of any event by counting how often it occurs among all samples. This estimated probability becomes more accurate with more samples.
5. **Consistency**: When we say an estimate is "≈" to a probability, it means the estimate becomes exact in the limit of infinitely many samples. This type of estimate is called consistent.

6. **Example**: For instance, if we sample 1000 times and find that 511 of those samples have Rain = true, then we estimate the probability of rain as 0.511.
7. These points highlight how sampling from a Bayesian network helps us understand probabilities of different events, even when the exact probabilities are not initially known.

Algorithm: Rejection sampling in Bayesian networks

Rejection sampling is a method used to estimate probabilities when dealing with uncertain situations, like predicting if it will rain based on certain observations. Here's a simplified explanation with examples:

What is rejection sampling?

Rejection sampling helps us estimate probabilities when direct calculation is hard. Imagine you want to know the chance of rain (R) given that the sprinkler is on (S = true), denoted as P(R | S = true).

How does rejection sampling work?

1. **Generate Samples**: First, generate many random scenarios (samples) of sprinkler and rain using what we know about their typical occurrences (prior knowledge).

   Example: Assume we know from past data:
   - P(Rain) = 0.3 (30% chance of rain on any given day)
   - P(Sprinkler) = 0.4 (40% chance the sprinkler is on on any given day)
2. **Apply Evidence**: Apply the evidence (observations) we have. In our example, we observe that the sprinkler is on (S = true).
   - Among the samples where S = true, count how many times it also rains (R = true).
3. **Estimate Probability**: Estimate P(R | S = true) based on the ratio of how often it rains when the sprinkler is on compared to all times the sprinkler is on.

   Example: Out of 100 days sampled where the sprinkler is on (S = true):
   - 40 days might have the sprinkler on (based on our prior)
   - Among those 40, we find it rains on 12 days.

   Therefore, P(R | S = true) ≈ 12/40 = 0.3.

Why is rejection sampling limited?

Rejection sampling becomes less effective as more variables (like more evidence to consider) are involved because it discards many samples that don't match the evidence. For instance, if we had to consider not just the sprinkler but also the color of the sky the previous night (another evidence), we might end up with very few samples that fit all conditions.

Real-world comparison:

In everyday life, estimating probabilities can be similar to rejection sampling. For example, to guess if it will rain (R) after seeing a red sky at night (RedSkyAtNight = true):

- We might observe how often it rains after seeing a red sky, ignoring days when the sky wasn't red.

  If we observe 50 red sky nights and it rains 15 times the next day, then P(R | RedSkyAtNight = true) ≈ 15/50 = 0.3.

Conclusion:

Rejection sampling provides a way to estimate probabilities in complex scenarios by generating and filtering scenarios based on observed evidence. However, it becomes impractical for highly complex problems due to the large number of samples it discards.

function REJECTION-SAMPLING(X, **e**, bn, N) **returns** an estimate of **P**(X|**e**)
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network
    N, the total number of samples to be generated
  local variables: **N**, a vector of counts for each value of 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**)

Figure 14.14 The rejection-sampling algorithm for answering queries given evidence in a Bayesian network.

Algorithm Explanation: Rejection sampling for Bayesian networks:

Rejection sampling is a Monte Carlo method used in Bayesian networks for generating samples from joint probability distributions:

1. **Basic Idea:** It randomly assigns values to variables based on priors and accepts or rejects samples based on observed evidence.
2. **Procedure:**
   a. Randomly assign values to variables according to priors.
   b. Check if the sample satisfies observed evidence; keep or reject it accordingly.
   c. Repeat until enough valid samples are collected.
3. **Sampling Efficiency:** Can be inefficient with low prior probabilities of satisfying evidence, leading to many rejections.
4. **Consistency:** Provides consistent estimates of the posterior distribution with sufficient samples, but can be computationally expensive for large networks or strict evidence.
5. **Suitability:** Easy to implement but may struggle with scalability in complex networks or dependencies.
6. **Comparison:** Less efficient compared to methods like likelihood weighting due to higher rejection rates, wasting computational resources.Rejection sampling ensures correct posterior estimates with enough samples, its inefficiency in managing evidence constraints and scalability issues restrict its practical utility in many Bayesian network scenarios.

1. **Input:**
    - X: Variable we want to estimate.
    - e: Evidence (observed values of variables).
    - bn: Bayesian network representing dependencies.
    - N: Number of samples to generate.
2. **Initialization:**
    - N: Vector to count occurrences of each value of X, initially all counts are zero.
3. **Procedure:**
    - Repeat N times:
        - Sample a configuration x from the prior distribution defined by bn.
        - If x matches the evidence e:
            - Increment the count corresponding to the value of X in x in vector N.
4. **Output:**
    - Normalize the counts in N to obtain an estimate of $P(X|e)$.

Explanation:

- The algorithm generates N samples from the Bayesian network's prior distribution.
- For each sample, it checks if it matches the given evidence e.
- If it matches, it updates the count of X's value in the sample.
- Finally, it normalizes these counts to produce the estimate of $P(X|e)$.

This method relies on generating samples and selecting only those consistent with the observed evidence, making it suitable for querying Bayesian networks where direct computation of probabilities may be impractical.

Algorithm: Likelihood Weighting:

- Likelihood weighting is a technique in artificial intelligence used for probabilistic inference.
- It works by sampling from the Bayesian network, focusing more on variables that are relevant to the query being evaluated.
- This method improves efficiency compared to exhaustive enumeration of all possible outcomes.
- Likelihood weighting efficiently computes probabilities in Bayesian networks by focusing sampling efforts on events consistent with observed evidence, and adjusting for the likelihood of those events given the evidence.
- This avoids the high rejection rates seen in rejection sampling, making it a more effective method for inference.

Likelihood Weighting Overview:

Likelihood weighting is a technique used in Bayesian networks to efficiently compute probabilities of queries given evidence. It avoids the inefficiencies of rejection sampling by only generating events that are consistent with the evidence provided.

Example Scenario:

Let's consider a simplified Bayesian network concerning weather conditions:

1. **Variables and Probabilities:**
    - **Cloudy (C)**: P(C=true) = 0.5, P(C=false) = 0.5
    - **Sprinkler (S)**: P(S=true | C=true) = 0.1, P(S=true | C=false) = 0.5
    - **Rain (R)**: P(R=true | C=true) = 0.8, P(R=true | C=false) = 0.2
    - **WetGrass (W)**: P(W=true | S=true, R=true) = 0.9, P(W=true | S=true, R=false) = 0.8, P(W=true | S=false, R=true) = 0.2, P(W=true | S=false, R=false) = 0.1
2. **Evidence:**
    - We have evidence that it is **Cloudy = true** and **WetGrass = true**.

3. **Query**:
    - We want to compute the probability of **Rain** = **true** given **Cloudy** = **true** and **WetGrass** = **true**, i.e., P(R=true | C=true, W=true).

Likelihood Weighting Process:

1. **Initialization**:
    - Start with a weight **w** = **1.0**.
2. **Sampling Process**:
    - Since **Cloudy** is evidence and fixed to true:
        - Set weight $w \leftarrow w \times P(C = true) = 0.5$.
    - Sample **Sprinkler** given **Cloudy** = **true**:
        - Suppose it returns **S=false** (since we sample from P(S=false | C=true) = 0.9).
    - Sample **Rain** given **Cloudy** = **true**:
        - Suppose it returns **R=true** (since we sample from P(R=true | C=true) = 0.8).
    - Since **WetGrass** is evidence and fixed to true:
        - Set weight $w \leftarrow w \times P(W = true|S = false, R = true) = 0.2$.
3. **Result of the Weighted Sample**:
    - The sampled event is [true, false, true, true] (Cloudy=true, Sprinkler=false, Rain=true, WetGrass=true) with a weight of 0.2.
4. **Tallying the Results**:
    - This event contributes to the count for **Rain** = **true**.

Why Likelihood Weighting Works:

- **Consistency with Evidence**: Likelihood weighting ensures that each sampled event is consistent with the given evidence (in this case, Cloudy=true and WetGrass=true).
- **Weighting by Likelihood**: The weight of each event is adjusted based on how likely it is under the evidence conditions. Events that match the evidence well receive higher weights, improving the accuracy of the probability estimation.

1. **Sampling with Likelihood Weighting**: In Bayesian networks, the Sampling with Likelihood Weighting (SWS) algorithm samples variables (including the query variable) given its parents' values.

2. **Influence of Evidence**: SWS considers evidence but less than the true posterior distribution. For instance, when sampling the Sprinkler variable, it takes into account evidence like "Cloudy = true" but ignores evidence from variables like "WetGrass = true."

3. **Weight Calculation**: Each sample's weight compensates for the discrepancy between the actual and desired sampling distributions. It's calculated based on the likelihoods of evidence variables given their parents.

4. **Weighted Probability**: The weighted probability of a sample combines the sampling distribution and the likelihood weights, ensuring that the overall probability estimate matches the joint probability distribution.

5. **Consistency**: Likelihood weighting estimates are consistent, meaning they converge to the true posterior probability with increasing sample size, as shown by the equation $\hat{P}(x|e) \approx P(x|e)$ for large N.

6. **Efficiency and Limitations**: Likelihood weighting is more efficient than rejection sampling because it uses all generated samples. However, its performance degrades with more evidence variables, as most samples end up with very low weights, skewing the weighted estimate.

7. **Impact of Variable Ordering**: If evidence variables appear late in the variable order, nonevidence variables may lack guidance from evidence in their ancestors, resulting in samples that don't reflect the real-world scenario indicated by the evidence.

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
    N, the total number of samples to be generated
  local variables: **W**, a vector of weighted counts for each value of X, initially zero

  for $j = 1$ to N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn, **e**)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in **x**
  return NORMALIZE(**W**)

---

function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight

  $w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    if X_i is an evidence variable with value x_i in **e**
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

Figure 14.15 The likelihood-weighting algorithm for inference in Bayesian networks. In WEIGHTED-SAMPLE, each nonevidence variable is sampled according to the conditional distribution given the values already sampled for the variable's parents, while a weight is accumulated based on the likelihood for each evidence variable.

1. **Objective**: Estimate $P(X \mid e)$, which is the probability of variable X given evidence e.

2. **Inputs**:

   - X: Query variable for which we want to estimate the probability.
   - e: Observed values of evidence variables E.
   - bn: Bayesian network that defines the joint distribution $P(X_1, \ldots, X_n)$.
   - N: Total number of samples to generate.

3. **Local variables**:

   - W: Vector of weighted counts for each value of X, initially set to zero.

4. **Algorithm Execution**:
    - **Initialization**: Start a loop that runs N times to generate samples.
    - **Weighted Sample Generation** (Function **WEIGHTED-SAMPLE**):
        - Initialize weight w to 1.
        - Initialize event x with n elements, using values from e.
        - For each variable X_i in $X_1, \ldots, X_n$:
            - If X_i is an evidence variable with a specified value x_i in e:
                - Update w by multiplying it with $P(X_i = x_i \mid \text{parents}(X_i))$.
            - Otherwise (if X_i is not evidence):
                - Sample a value for X_i from $P(X_i \mid \text{parents}(X_i))$.
        - Return the event x and the weight w.
    - **Weight Accumulation**: After generating each sample in **WEIGHTED-SAMPLE**:
        - Update $W[x]$ by adding w, where x is the value of X in the current sample x.
    - **Normalization**: Once all N samples are generated and weights accumulated:
        - Normalize the vector W to get an estimate of $P(X \mid e)$.
5. **Output**: Return the normalized vector W, ↓ ich provides an estimate of $P(X \mid e)$.

Summary: The algorithm generates samples from the Bayesian network conditioned on the evidence e. For each sample, it calculates a weight based on how consistent the sample is with the observed evidence. After generating N such samples, it estimates $P(X \mid e)$ by normalizing the accumulated weights of the samples where X takes each possible value.

Algorithm-4: Inference by Markov chain simulation

MARKOV CHAIN MONTECARLO Markov chain Monte Carlo (MCMC)algorithms

Markov chain Monte Carlo (MCMC) algorithms work differently than methods like rejection sampling and likelihood weighting. Instead of creating each sample from scratch, MCMC generates samples by tweaking the previous sample randomly.

Think of MCMC as being in a current state, where every variable has a value. It moves to a next state by making random changes to the current state. This process is akin to how simulated annealing or WALKSAT work, which are also types of MCMC methods.

One specific type of MCMC is Gibbs sampling, which is great for Bayesian networks. Here's how it works:

Imagine you have variables X and Y, and you want to sample from their joint distribution. Gibbs sampling does this by iteratively sampling each variable conditioned on the current values of the others. For instance, if you know the value of Y, you can sample X based on the distribution of X given Y. Then, you update Y based on the new value of X, and repeat.

This method helps in exploring complex distributions where directly sampling is hard, by gradually moving through states that approximate the desired distribution.

Gibbs sampling in Bayesian networks

- In Gibbs sampling for Bayesian networks, we start with an initial guess where observed variables like 'Sprinkler' and 'Wet Grass' are fixed. We then randomly set the values of other variables, such as 'Cloudy' and 'Rain'. For example, let's say we start with 'Cloudy' as true and 'Rain' as false: [true, true, false, true].
- The algorithm then updates one variable at a time based on its Markov blanket—its parents, children, and children's parents. This process continues, flipping variables while keeping the observed variables fixed.
- This random wandering through possible assignments helps us estimate probabilities like P(Rain | Sprinkler = true, Wet Grass = true)

Now the nonevidence variables are sampled repeatedly in an arbitrary order. For example:

1. **Sampling Cloudy:**
   - Imagine we're trying to determine if it's cloudy outside. We look at related factors, like if the sprinkler is on and if it's raining.

- Suppose we know the sprinkler is on and it's not raining. Based on our model, if we find that the probability of cloudy being false is higher, we update our current situation accordingly.

2. **Sampling Rain:**

- Next, let's determine if it's raining. This depends on factors like whether it's cloudy, if the sprinkler is on, and if the grass is wet.
- For instance, if we know it's not cloudy, the sprinkler is on, and the grass is wet, we might find that rain is likely to be true.

3. **Counting States:**

- As we go through these steps, each combination of conditions we consider forms a state. Some states will have rain as true, others as false.
- Let's say we count 20 states where rain is true and 60 states where it's false.

4. **Calculating the Probability:**

- To answer how likely it is to rain, we normalize the counts we found. This means dividing the number of true rain states by the total number of states considered.
- In our example, with 20 true rain states and 60 false ones, the normalized probability is 0.25 (or 25%) for rain being true and 0.75 (or 75%) for rain being false.
- In Bayesian networks help us compute probabilities by sampling different scenarios based on known conditions, ultimately giving us a clearer picture of the likelihood of events like rain occurring.

The complete algorithm is shown in Figure 14.16.

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **e**

  initialize **x** with random values for the variables in **Z**
  for $j = 1$ to N **do**
    for each Z_i in **Z do**
      set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

The Gibbs sampling algorithm for approximate inference in Bayesian networks:

The Gibbs sampling algorithm is used for estimating probabilities in Bayesian networks when some evidence about variables is known.

1. **Initialize**: Start with a vector N to count occurrences of each value of X. Also, identify the variables Z in the Bayesian network that are not part of the evidence e. Copy the evidence into x, the current state of the network.

2. **Random Initialization**: Assign random values to the variables in Z that are not in the evidence e .

3. **Sampling Process**:

   - **Iterations**: Repeat the following steps N times (where N is a parameter):

   - **For each variable Zi in Z**:

     - Set Zi's value in x by sampling from its conditional probability $P(Zi \mid \text{mb}(Zi))$.

       - $\text{mb}(Zi)$ denotes the Markov blanket of Zi, which includes its parents, children, and children's other parents.

   - **Counting**: After updating x with the new sampled values, increment the count $N[x]$, where x is the current state of X in x.

4. **Normalization**: Once all iterations are completed, normalize the counts N to get an estimate of $P(X \mid e)$.

Return: Finally, return the normalized counts N, which represent the estimated probabilities $P(X \mid e)$.

This algorithm essentially simulates the distribution of variables in the Bayesian network, focusing on the variables not given as evidence and updating their values based on their neighbors' influences. Through repeated sampling and counting, it provides an approximate inference of the probability distribution of interest.

Markov chains, stationary distributions, and detailed balance:

1. **Markov Chain Basics**:
   - A Markov chain is defined by transition probabilities $q(x \to x')$, which describe the likelihood of moving from state x to state x'.
2. **Stationary Distribution $\pi(x)$**:
   - $\pi_t(x)$ is the probability of being in state x at time t.
   - The chain reaches its stationary distribution π when $\pi_t = \pi_{t+1}$.
   - Stationarity is described by $\pi(x) = \pi(x)q(x \to x)$ for all x.
3. **Detailed Balance (Equation 14.11)**:
   - Detailed balance implies that $\pi(x)q(x \to x') = \pi(x')q(x' \to x)$ for all x and x'.
   - This condition ensures that the flow (expected transition) between any pair of states is balanced in both directions.
4. **Relationship Between Detailed Balance and Stationarity**:
   - Detailed balance (Equation 14.11) implies stationarity (Equation 14.10).
   - Stationarity means the system's distri ↓ ion does not change over time; it's in equilibrium.
5. **Ergodicity**:
   - The system is ergodic if every state is reachable from every other state and there are no strictly periodic cycles.
   - For ergodic Markov chains with a unique stationary distribution π, detailed balance condition holds.

In summary, detailed balance condition $\pi(x)q(x \to x') = \pi(x')q(x' \to x)$ ensures that the Markov chain has a unique stationary distribution π, where $\pi(x)$ represents the long-term probability of being in state x. This condition is fundamental in understanding the equilibrium properties of Markov chains.

Gibbs sampling is a method used in Bayesian networks to generate samples from the posterior distribution of variables given evidence:

1. **Gibbs Sampling Overview**:
   - Gibbs sampling iterates through each variable in the network, updating it based on the current values of all other variables (including evidence).
   - It ensures detailed balance, meaning the sampling process reaches a stationary distribution equal to the true posterior distribution $P(x|e)$.
2. **Transition Probability**:
   - Each variable X_i is updated based on its Markov blanket (MB), which includes its parents and children in the Bayesian network.
   - For example, if X_i has parents $\text{Parents}(X_i)$ and children $\text{Children}(X_i)$, the conditional probability is proportional to:

$$P(x_i|\text{mb}(X_i)) = \alpha \cdot P(x_i|\text{Parents}(X_i)) \times \prod_{Y_j \in \text{Children}(X_i)} P(y_j|\text{Parents}(Y_j))$$

3. **Detailed Balance**:
   - Gibbs sampling maintains detailed balance by ensuring:

$$\pi(x) q_i(x \to x) = P(x|e) q_i(x \to x)$$

     where $\pi(x)$ is the true posterior and $q_i(x \to x)$ is the transition probability for variable X_i.
4. **Ergodicity and Stationarity**:
   - If the conditional probability distributions q_i for each variable maintain the stationary distribution $P(x|e)$ and the network is connected (no probabilities of 0 or 1 causing disconnects), Gibbs sampling is ergodic. This means it eventually samples from $P(x|e)$.
5. **Implementation**:
   - To perform Gibbs sampling:
     - Start with initial values for all variables.
     - Sequentially update each variable X_i using its Markov blanket until convergence.
     - Repeat until enough samples are obtained from $P(x|e)$.

9th Topic: Relational and First-Order Probability Models

I. Representational Advantages of First-Order Logic vs. Bayesian Networks

In Chapter 8, explored how first-order logic improves upon propositional logic by allowing more concise expression of relationships among objects and properties. This is crucial in domains with varying numbers of objects and complex interactions.

Limitations of Bayesian Networks

Bayesian networks are constrained by their propositional nature, limiting them to fixed sets of variables and predefined value domains. This can restrict their applicability in scenarios with nuanced interactions.

Example Scenario: Assessing Book Quality with Customer Recommendations

Consider an online book retailer analyzing book quality based on customer recommendations:

- **Bayesian Networks Approach**: A basic method might average recommendations, adjusting for variance based on the number received. This overlooks factors like varying customer honesty and kindness, which can bias recommendations.
- **First-Order Logic Approach**: Here, each customer's recommendation for a book (e.g., recommending Book A to Customer X) can be represented with variables such as Honest(X), Kindness(X), and Quality(Book A). Rather than manually specifying these relationships repeatedly, first-order logic uses a structured approach where the recommendation (Recommendation(X, Book A)) depends on these variables.

Example in First-Order Logic:

```
plaintext
Copy code
Recommendation(X, Book A) ~ RecCPT(Honest(X), Kindness(X), Quality(Book A))
```

This states that the recommendation of Book A by Customer X follows a Conditional Probability Table (CPT), which considers the variables Honest(X), Kindness(X), and Quality (Book A).

Benefits of First-Order Logic:

- **Flexibility**: It allows for a more flexible representation of complex interactions among customers and books.
- **Scalability**: Easily scales with increasing numbers of customers and books without the need for repetitive manual adjustments to the network structure.

- **Nuanced Analysis**: Enables a nuanced analysis by capturing diverse customer behaviors and book attributes that a fixed Bayesian network structure may struggle to accommodate.
- Bayesian networks are useful for certain structured problems, first-order logic provides a more adaptable framework for modeling dynamic and intricate relationships in domains where such variability is significant.

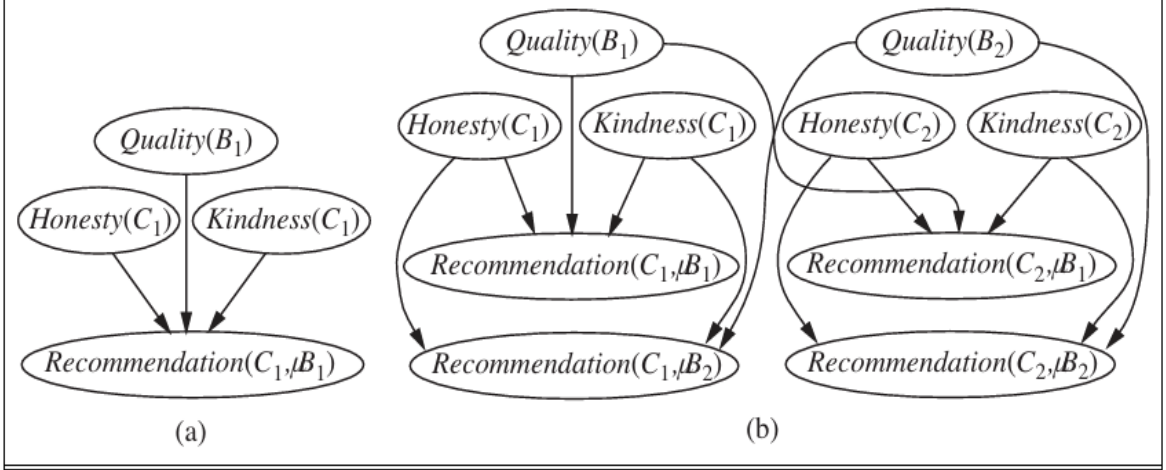

Figure 14.17 (a) Bayes net for a single customer C_1 recommending a single book B_1. $Honest(C_1)$ is Boolean, while the other variables have integer values from 1 to 5. (b) Bayes net with two customers and two books.

II. Possible worlds

1. **Possible Worlds in Probability Models:**
    - In probability theory, a set of possible worlds, denoted as Ω, represents all conceivable scenarios or states of affairs.
    - Each world ω in Ω has an associated probability P(ω), which tells us how likely that particular world is.

    Example: Consider a coin toss where Ω = {Heads, Tails}. If we assume a fair coin, P(Heads) = P(Tails) = 0.5.
2. **Possible Worlds in Bayesian Networks:**
    - In Bayesian networks, possible worlds are specific assignments of values to variables that satisfy the network's structure and conditional probabilities.

    Example: Imagine a Bayesian network for diagnosing diseases, where variables could be {Symptoms, Disease}. Possible worlds would be {Fever, Flu} or {No Fever, No Flu}, etc.

3. **First-Order Probability Models:**
    - For first-order logic, possible worlds involve a richer structure. They encompass sets of objects with relations among them, akin to the semantics of first-order logic.

    Example: Suppose we have a domain with objects {Alice, Bob} and a relation Likes(x, y) which denotes x likes y. Possible worlds would include assignments such as {Likes(Alice, Bob)} or {Likes(Bob, Alice)}.
4. **Challenges with Infinite Sets of Worlds:**
    - One issue with first-order models is that the set of possible worlds can be infinite. For instance, if we have an infinite number of individuals (objects) and relations, the number of possible worlds becomes uncountably large.
5. **Dealing with Infinite Sets:**
    - To manage this, one approach is to adopt database semantics, where we assume a finite set of possible worlds by restricting the domain to named constants (unique names assumption) and ensuring domain closure (no unnamed objects).

    Example: In a domain of books and readers, if we only consider named readers {Alice, Bob} and books {Book1, Book2}, possible worlds are constrained to combinations like {Alice likes Book1, Bob likes Book2}.

The first-order logic allows for rich modeling of complex scenarios, managing infinite sets of possible worlds poses challenges. Techniques like database semantics help in restricting these sets to manageable sizes for probabilistic reasoning.

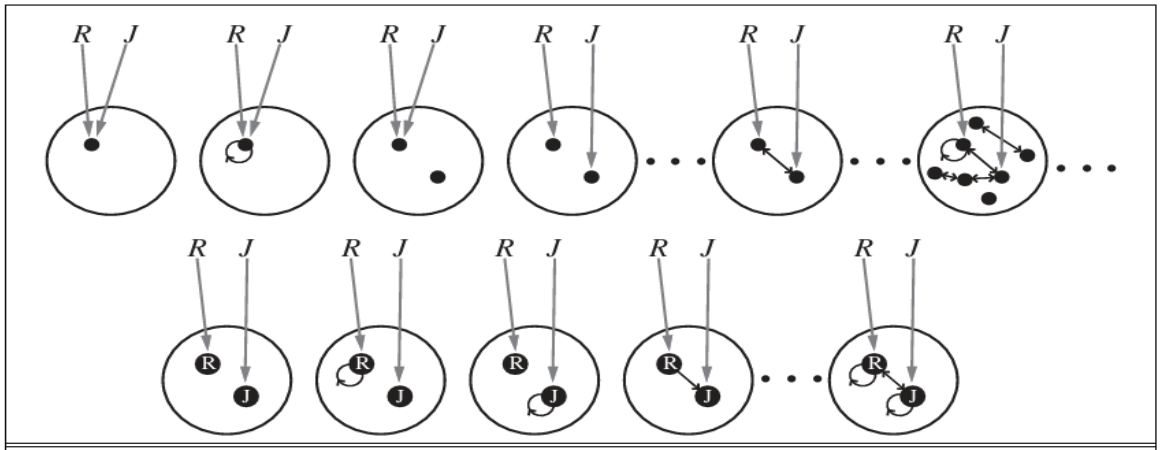

Figure 14.18 Top: Some members of the set of all possible worlds for a language with two constant symbols, R and J, and one binary relation symbol, under the standard semantics for first-order logic. Bottom: the possible worlds under database semantics. The interpretation of the constant symbols is fixed, and there is a distinct object for each constant symbol.

Relational Probability Model

SIBYL addresses uncertainties in symbolically represented object existence and identity.

SIBYL addresses uncertainties regarding the existence of objects behind observed symbols and whether different symbols refer to the same object.

In a Relational Probability Model (RPM), we deal with uncertainty about how symbols correspond to real-world objects. Unlike traditional databases that assume everything not known is false, RPMs acknowledge uncertainties like whether different symbols refer to the same object or how many objects exist.

For example, imagine a book retailer using ISBNs to identify books. A single book like "Gone With the Wind" might have multiple ISBNs, which complicates tracking recommendations or sales across different identifiers. Similarly, customers using multiple login IDs can confuse systems meant to track reputation, like in online reviews or security systems.

These uncertainties are critical because:

1. **Existence Uncertainty:** We're unsure about which objects truly exist behind the symbols we observe. For instance, how many distinct copies of "Gone With the Wind" are there?
2. **Identity Uncertainty:** We may not be certain which symbols actually refer to the same underlying object. For example, are two different ISBNs really identifying the same physical book?

RPMs help us model scenarios where what we observe (like ISBNs or customer IDs) might not perfectly reflect the underlying reality (actual books or unique customers). This flexibility in handling uncertainty makes RPMs useful in areas like online retail and security where clear identifications can be tricky due to various factors.

Relational Probability Models

Relational Probability Models (RPMs) are similar to first-order logic but use symbols for constants, functions, and predicates. Predicates act like functions that determine true or false statements.

In the context of recommending books to customers, here are the key elements simplified:

1. **Symbols and Types:**
    - **Constants**: Names of customers (like C1, C2) and books (like B1, B2).
    - **Functions** (returning values):
        - **Honest**: Determines if a customer is honest (`Honest(C1)`).
        - **Kindness**: Rates the kindness of a customer (`Kindness(C1)`).

- **Quality**: Rates the quality of a book (`Quality(B1)`).
- **Recommendation**: Rates how likely a customer recommends a specific book (`Recommendation(C1, B1)`).

Each function has specific types for its arguments and return values:

- `Honest` and `Kindness` are boolean (true/false).
- `Quality` is rated on a scale of 1 to 5.
- `Recommendation` is rated on a scale of 1 to 5.

2. **Random Variables:**
   - RPM defines random variables by instantiating each function with specific combinations of customers and books. For instance:
     - `Honest(C1)`, `Honest(C2)`, etc.
     - `Quality(B1)`, `Quality(B2)`, etc.
     - `Recommendation(C1, B1)`, `Recommendation(C1, B2)`, etc.
3. **Dependencies:**
   - Each function has dependencies that govern its behavior. These are statements about the likelihood of certain values based on other factors:
     - `Honest(c)` might have probabilities like 0.99 for true and 0.01 for false.
     - `Kindness(c)` might have probabilities like 0.1, 0.1, 0.2, 0.3, 0.3 for its ratings.
     - `Quality(b)` might have probabilities like 0.05, 0.2, 0.4, 0.2, 0.15 for its ratings.
     - `Recommendation(c, b)`'s probability might be derived from a conditional distribution table (`RecCPT`) based on the honesty, kindness, and quality values associated with `c` and `b`.
4. **Bayesian Network:**
   - By instantiating these dependencies for all known customers and books, RPM forms a Bayesian network.
   - This network defines a joint probability distribution over all the random variables, representing how likely different states (like honesty, kindness, quality, recommendation) are given the relationships between customers and books.

RPMs help model relationships and probabilities between entities like customers and books, enabling systems to make informed predictions and recommendations based on data and logical dependencies defined for each entity.

Context Specific Independence

Context-Specific Independence

Imagine a scenario where customers give recommendations for books. Some customers are honest, while others might not be truthful in their reviews. Dishonest customers might ignore a book's quality when recommending it, and their kindness doesn't affect their decisions.

Conditional Dependence Example

1. **Basic Independence Example:**
   - If a customer c is dishonest ($Honest(c) = \text{false}$), their recommendation ($Recommendation(c, b)$) doesn't depend on how kind they are ($Kindness(c)$) or the book's quality ($Quality(b)$).
   - For honest customers ($Honest(c) = \text{true}$), their recommendation depends on both kindness and quality.
2. **Adding Fan of an Author:**
   - An honest customer who is a fan of a specific author ($Fan(c, \text{Author}(b))$) always gives the author's books a perfect rating of 5, regardless of the book's quality.
   - Otherwise, they use a more nuanced recommendation rule ($HonestRecCPT(Kindness(c), Quality(b))$).

Handling Uncertainty

1. **Unknown Author Example:**
   - If we don't know who the author of a book $B2$ is ($\text{Author}(B2)$), we might have to consider all possible authors, say $A1$ and $A2$.
   - Each potential author (like $A1$ or $A2$) could be favored by a customer ($Fan(C1, A1)$, $Fan(C1, A2)$), influencing their recommendation.
2. **Relational Uncertainty:**
   - The uncertainty about $\text{Author}(B2)$ affects how the recommendation system works. If three customers are all fans of $A1$ and have given $B2$ a perfect score, it strongly suggests that $A1$ is likely the author of $B2$.

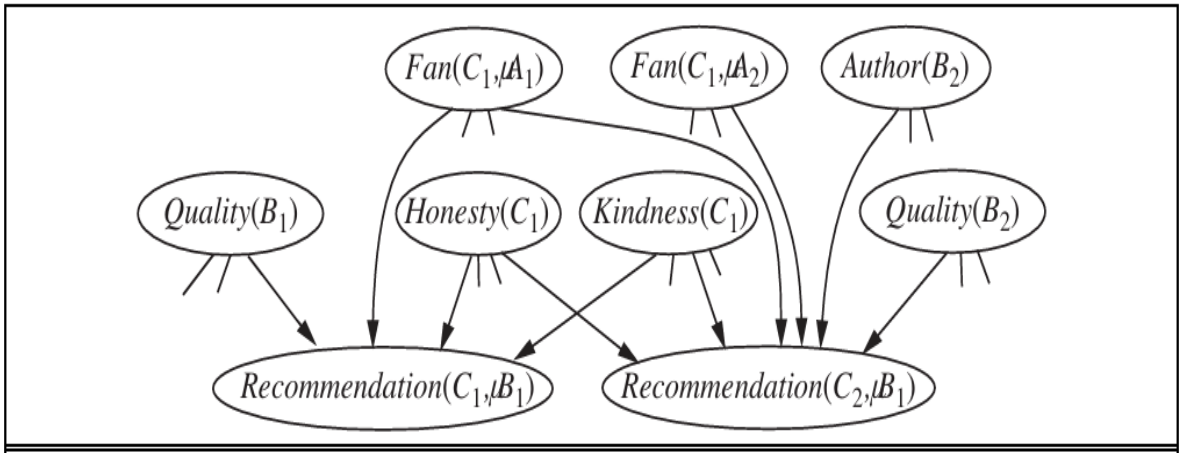

Figure 14.19 Fragment of the equivalent Bayes net when $Author(B_2)$ is unknown.

1. **Emergence of Sophisticated Reasoning**: RPM models demonstrate how probabilities spread through connections among objects, leading to clearer and more reliable results as connections and objects increase.
2. **Inference in RPMs**: "Unrolling" involves gathering evidence and queries about constant symbols to construct an equivalent Bayes network for inference.
3. **Drawbacks of Unrolling**: The resulting Bayes network can become very large, especially with many possible objects related to unknown functions or relations.
4. **Improving Standard Inference Methods**: Techniques like caching can speed up computations by exploiting repeated structures in the unrolled Bayes network.
5. **Adapting Methods for RPMs**: Methods that exploit specific contexts and MCMC algorithms are beneficial for handling relational uncertainty in RPMs.
6. **MCMC Algorithm**: Samples potential scenarios where relational structures are fully understood, simplifying dependencies for each scenario and managing relational uncertainty effectively.
7. **Comparison with Logical Inference**: RPMs require partial unrolling into Bayesian networks similar to how logical propositions are handled in first-order logical inference.
8. **Alternative Approaches**: Systems like resolution theorem provers and logic programming avoid exhaustive unrolling by instantiating logical variables as needed, akin to the lifting approach in probabilistic inference.

Enhancing Book Rating Predictions: Managing Author Uncertainty and Customer Bias

Imagine a website where customers rate books. Some users are honest in their reviews, while others might not be truthful. Dishonest reviewers ignore a book's quality and their kindness doesn't influence their ratings.

For example:

- An honest customer might always give books by their favorite author a perfect rating of 5.
- If we don't know who wrote a certain book, we have to consider all possible authors and how much customers like each author to predict their ratings accurately.

This system deals with uncertainty about book authors and customer preferences to make better predictions about book ratings.

Open-universe probability models (Managing Ambiguity in Real-World Object Identification)

1. Unlike in databases with unique IDs, real-life systems like cameras, text processors, and intelligence analysis deal with ambiguity.
2. Cameras may not recognize if the object around a corner is the same as seen earlier.
3. Text processors must determine if different mentions (e.g., "Mary," "Dr. Smith," "she") refer to the same entity.
4. Intelligence analysts track spies without knowing how many there are or if various identifiers belong to the same spy.
5. Human understanding involves learning what objects exist and linking observations without clear, unique identifiers.
6. Dealing with uncertainty is a crucial challenge in these contexts.

OPEN UNIVERSE

Open-Universe Probability Models (OUPMs) compared to Bayesian Networks and their application in a book-recommendation context:

1. **Purpose**: OUPMs maintain a consistent probability distribution across all scenarios by allowing creation of new objects.
2. **Comparison with Bayesian Networks**:
    - OUPMs create and define distributions over new objects dynamically.
    - Bayesian networks use predefined structures to assign values to variables.
3. **Generating Objects**:
    - OUPMs model scenarios with variable numbers and types of objects.
    - Example: Predicting customer count and their login IDs using probabilistic distributions.
4. **Example in Book-Recommendation Domain**:
    - OUPMs predict customer numbers via distributions like log-normal.

- They differentiate between honest and dishonest customers with distinct distribution patterns for login IDs.

5. **Overall Benefit**: OUPMs offer a robust framework for probabilistic modeling by ensuring consistent distributions across all possible object configurations.

Customer Behavior Modeling with Log-Normal Distributions:

1. We expect between 100 and 10,000 customers, represented by a log-normal distribution with parameters $\mu = 6.9$ and $\sigma = 2.32$.
2. For honest customers, each customer has exactly one ID.
3. For dishonest customers, the number of IDs follows a log-normal distribution with the same parameters $\mu = 6.9$ and $\sigma = 2.32$.

Origin Function:

In BLOG, imagine a system where each customer has several unique login IDs. Each possible scenario or "world" in this system shows exactly how these IDs were created. To ensure accuracy, the system follows specific rules to calculate probabilities across all these scenarios. Algorithms in BLOG can figure out answers to questions about these scenarios, getting closer to the true likelihoods over time. It's tricky to design these algorithms, especially when dealing with endless possibilities. Despite these challenges, using first-order logic for probabilistic reasoning significantly improves AI's ability to handle uncertain information, like in computer vision or intelligence analysis.