# Overview

I Artificial Intelligence

II Problem Solving

III Knowledge, Reasoning, Planning

**IV Uncertain Knowledge and Reasoning**

12. Quantifying Uncertainty
13. Probabilistic Reasoning
14. Probabilistic Reasoning over Time
15. **Probabilistic Programming**
16. Making Simple Decisions
17. Making Complex Decisions
18. Multiagent Decision Making

V Machine Learning

VI Communicating, Perceiving, and Acting

VII Conclusions

# Probabilistic Programming

- Basic problem: Increasing the expressiveness of probability models from factored to structured models
  - Bayesian networks reflect propositional logic
  - Two roads: Using relational models (equivalent to first order logic) and via traditional programming lanuages

Content:

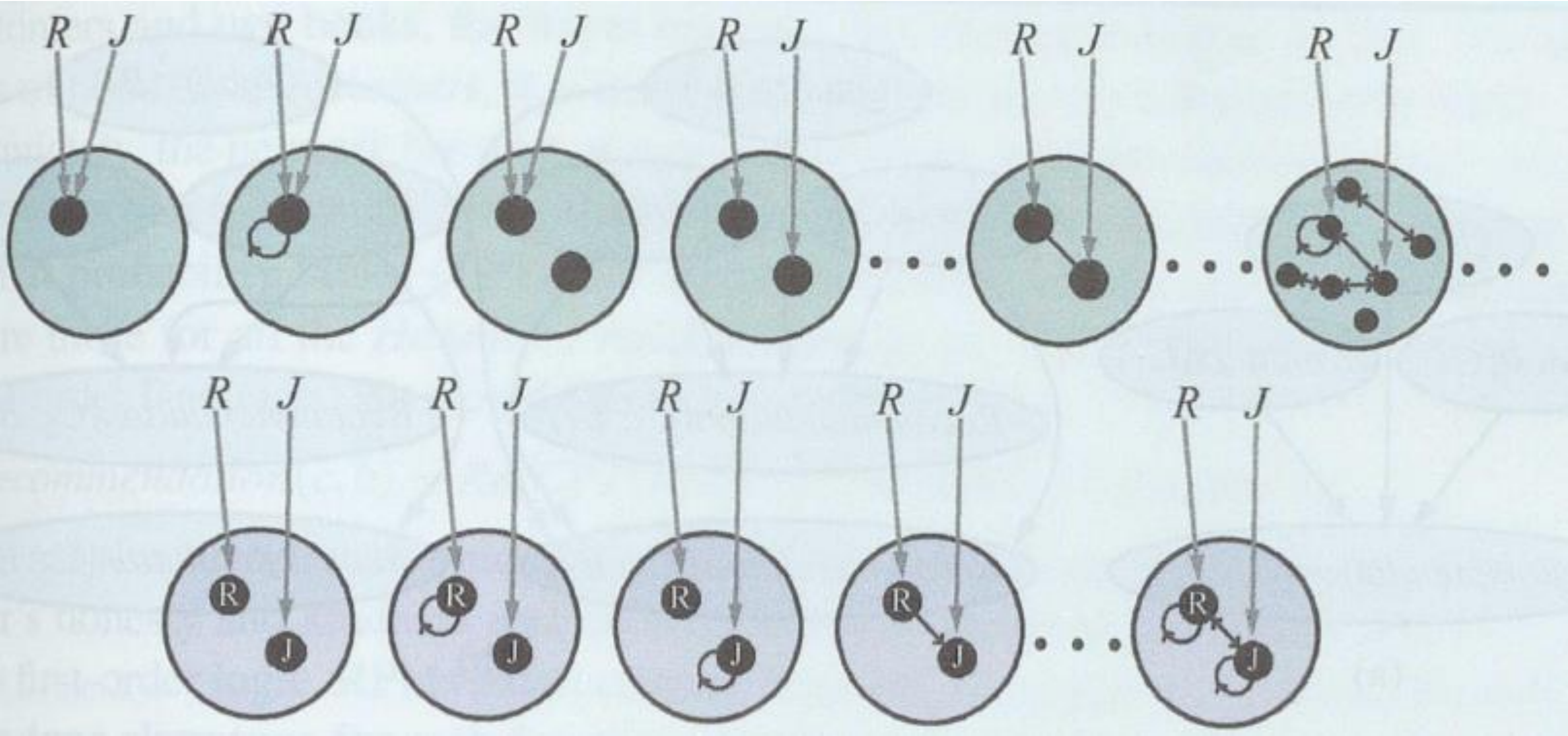
- Relational Probability Models
- Open-Universe Probability Models
- Keeping Track of a Complex World
- Programs as Probability Models

# Relational Probability Models (1)

- Idea: First-order probability models
  - Assigning possible worlds to a set of objects with relations among them and mapping constant symbols to objects, predicate symbols to relations and function symbols to functions on those objects.
  - Then, the probability of a first oder logical sentence $\phi$ is the sum over the possible worlds where it is true: $P(\phi) = \sum_{\omega:\ \phi \text{ is true in } \omega} P(\omega)$

# Relational Probability Models (2)

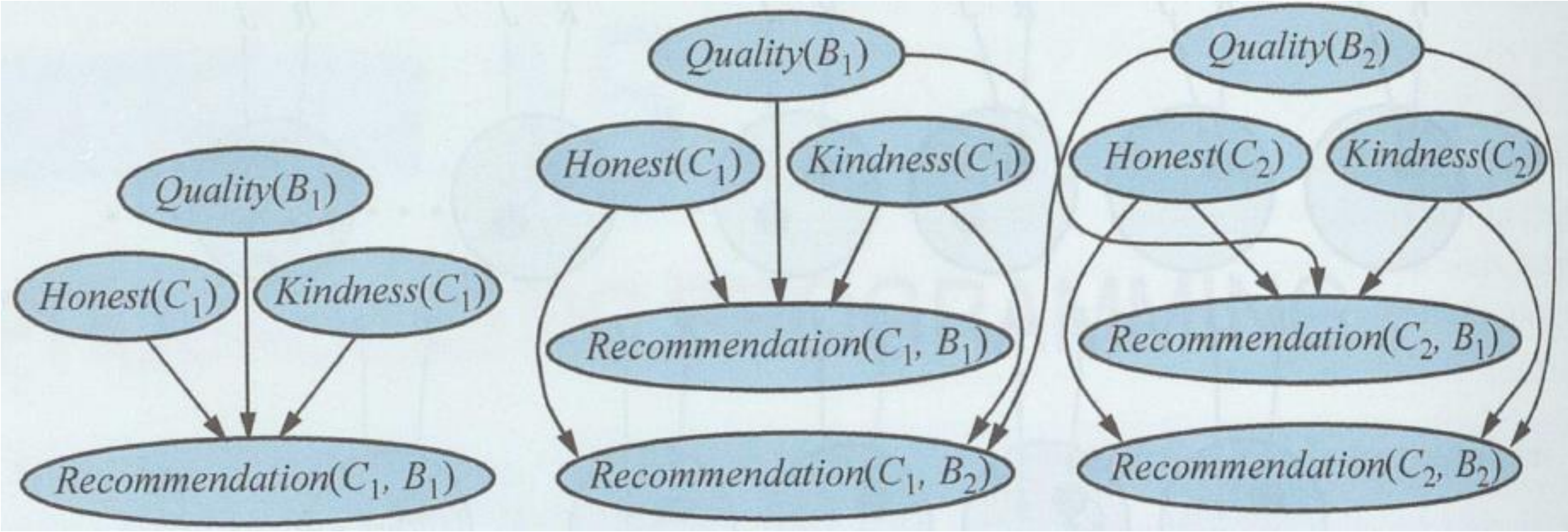
- Problem: The set of first-order models is infinite
  - Example: Two constant symbols and one binary relation symbol (top row)
- Solution: Use database semantics (second row)
  - Unique name assumption
  - Domain closure (guarantees a finite set of possible worlds)
  - *No* closed world assumption




# Syntax and Semantics of RPMs



- RPM = Relational Probability Model
- Example: Book evaluation based on book recommendations by customers
  - Simplest solution: Use average recommendations
  - Refined solution: Take into account, that some customers are kinder than others and some are dishonest (maybe paid for doing so) and refine average recommendations
    - Example Bayes net for one and two customers (C) for one or two books (B):
    - For larger networks we need RPMs!




# RPM for Book Recommendation

- Recommendation (c,b) $\sim$ RecCPT (Honest(c), Kindnes (c), Quality (b))
  - Recommendation conditional probability table (RecCPT) needs to specifiy all combinations of the values of its variables
  - Values given by a type signature for the functions and predicates:
    - Honest: Customer $\rightarrow$ {true, false}
    - Kindness: Customer $\rightarrow$ {1, 2, 3, 4, 5}
    - Qualitiy: Book $\rightarrow$ {1, 2, 3, 4, 5}
    - Recommendation: Customer x Book $\rightarrow$ {1, 2, 3, 4, 5}
  - Prior probabilities for the random variables:
    - Honest (c) $\sim \langle 0.99, 0.01 \rangle$
    - Kindness (c) $\sim \langle 0.1, 0.1, 0.2, 0.3, 0.3 \rangle$
    - Quality (b) $\sim \langle 0.05, 0.2, 0.4, 0.2, 0.15 \rangle$
    - RecCPT for Recommendation (c,b) has 2 x 5 x 5 = 50 rows, each with 5 entries

# Conditional Probability Tables



- RecCPT for Recommendation (c,b) with 250 parameters (5*50) is large, but much smaller than for an equivalent Bayesian net
  - Can be further reduced by context-specific indepence rules, e.g. dishonest customers ignore quality and kindness:
    - Recommendation (c, b) ~ **if** Honest (c) **then** HonestRecCPT (Kindness(c), Quality (b))
      **else** $\langle 0.4, 0.1, 0.0, 0.1, 0.4 \rangle$
  - Many more refinements possible, e.g. an honest customer who is fan of the book‘s author gives the book a „5“ regardless of quality:
    - Recommendation (c, b) ~ **if** Honest (c) **then**
      **if** Fan (c, Author(b)) **then** Exactly (5)
      **else** HonestRecCPT (Kindness(c), Quality (b))
      **else** $\langle 0.4, 0.1, 0.0, 0.1, 0.4 \rangle$
    - The conditional tests maybe unknown, but can be probabilistically infered from the recommendation database

# Example: Rating Player Skill Levels

- Many competitive games have a rating for players‘ skill level, e.g. Elo rating for chess (beginner: around 800, world champion around 2800).
- We can develop a Bayesian rating scheme with the following functions and predicates:
  - Each player i has a skill level: Skill (i)
  - In each game g, i‘ performance is: Performance (i, g)
  - Corresponding RPM:
    - Skill (i) ~ Normalverteilung ($\mu$, $\sigma^2$)
    - Performance (i,g) ~ Normalverteilung (Skill (i), $\beta^2$)   // $\beta^2$ variance of players actual performance
    - Win (i, j, g) = **if** Game (g, i, j) **then** Performance (i, g) > Performance (j, g)

# Inference in RPMs

- Most straightforward approach: Construct the equivalent Bayesian network
  - Use known constant symbols belonging to each type
  - Example of construction of the net for recommender model:

**for** $b = 1$ **to** $B$ *do*
  add node $Quality_b$ with no parents, prior $\langle 0.05, 0.2, 0.4, 0.2, 0.15 \rangle$
**for** $c = 1$ **to** $C$ *do*
  add node $Honest_c$ with no parents, prior $\langle 0.99, 0.01 \rangle$
  add node $Kindness_c$ with no parents, prior $\langle 0.1, 0.1, 0.2, 0.3, 0.3 \rangle$
  **for** $b = 1$ **to** $B$ *do*
    add node $Recommendation_{c,b}$ with parents $Honest_c, Kindness_c, Quality_b$
      and conditional distribution $RecCPT(Honest_c, Kindness_c, Quality_b)$

  - Technique is called **grounding** or **unrolling** (like propositionalization of first-order logic)

# Improvements for Unrolling

- Problem: The generated Bayesian net can become very large
- Improvements:
  - Avoid generating the full implicit net
    - Variables are irrelevant, if they are not an ancestor of a query or an evidence variable
    - Based on the values of the evidence variables, further variables can becomoe conditionally independent on the query variable
    - ➢ Instantiate only relevant variables when unrolling the net
  - Cache results for repeated substructures in the unrolled Bayes net
  - MCMC inference algorithm works an sampling complete possible worlds, where all variables are known. It can therefore deal efficiently with relational uncertainty, where e.g. the author of a book is not known, because in a sample, a concrete author is choosen.
  - In some cases, it is even possible to avoid unrolling the model althogether (implementation rather complicated)

# Open Universe Probability Models

- Database semantics is problematic in many use cases
  - Existence uncertainty and identity uncertainty, e.g.
    - Recommender model: Each book has an unique ISBN, but a „logical“ book may have different ISBNs for hardcover, paperback, large print, reissues etc. which should be aggregated, but that might be difficult. Similar, dishonest customers may use different IDs.
    - Vision system: If looking around a corner, are there some objects it has seen already?
    - Text understanding: Coreference resolution: Refer different phrases like „Mary“, „Dr. Smith“, „she“, „his mother“ etc. to the same entity?
    - Spy hunting
  - **Open Universe Probability Models (OUPM)** based on standard semantics of first order logic necessary

# OUPM

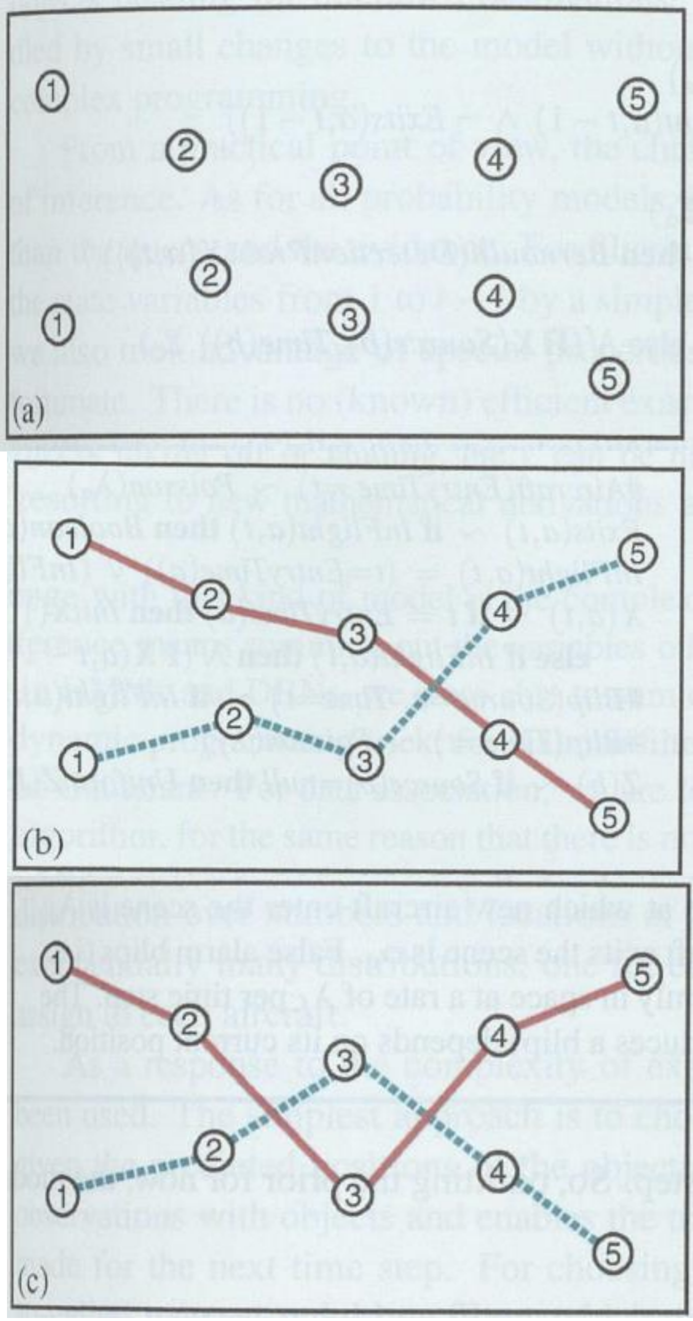
- Syntax and semantics of OUPM more complex than in RPMs
- Typical OUPMs have possible worlds of infinite size
  - Sampling for partial worlds necessary (for the relevant variables including their parents)
    - This restriction is used in RPMs for effiency reasons too
- Inference algorithm the same: MCMC, since it works entirely local
- Examples for Open Universe Probability Models:
  - Citation matching: Refer different citations to the same reference?
    - Much better than e.g. CiteSeer

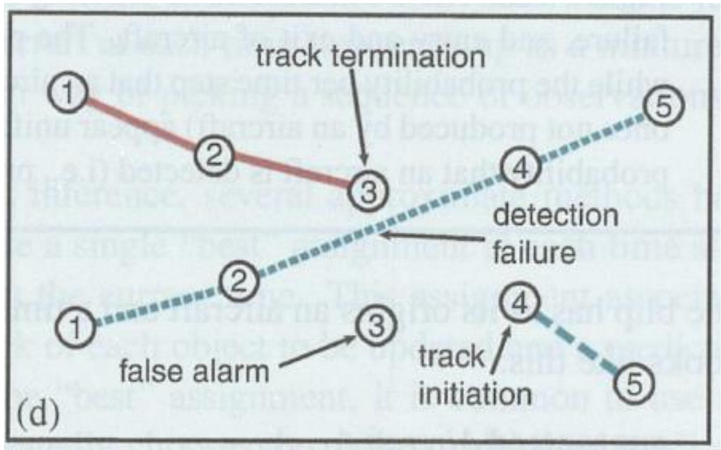
1. [Lashkari et al 94] Collaborative Interface Agents, Yezdi Lashkari, Max Metral, and Pattie Maes, Proceedings of the Twelfth National Conference on Articial Intelligence, MIT Press, Cambridge, MA, 1994.
2. Metral M. Lashkari, Y. and P. Maes. Collaborative interface agents. In Conference of the American Association for Artificial Intelligence, Seattle, WA, August 1994.

  - Nuklear treaty monitoring: NET-VISA since 2018 part of UN CTBTO monitoring system

# Keeping Track of a Complex World

- Problem: Keeping track of several moving objects
  - e.g. blinking objects on a radar screen (representing aircrafts)
  - (right): given „blips“ as in picture (a), is (b) or (c) a better hypothesis?
  - **Identity uncertainty** in a temporal context („**data association problem**“)
- Simplifying assumption: There are exactly n objects (e.g. aircrafts)
  - For T time steps, there are already $(n!)^T$ possible assignments
- General case: False alarms (blips caused by no object), detection failures, new objects can arrive and old ones disappear




# Algorithms for Keeping Track of a Complex World

- No efficient exact algorithm available
- Several approximate methods:
  - Simplest approach: Choose a single best assignment at each time step by e.g. **nearest-neighbor filter**: Closest pairing of predicted position and observation
  - Check all possible assignments by an assignment algorithm, e.g. Hungarian algorithm
- Problem:
  - Wrong assignment in one time step causes follow up errors in the following time steps
- Alternative solutions:
  - Sampling approaches like **particle filtering** for data association by maintaining large collections of possible current assignments
  - Combination of sampling approaches with exact inference (Rao-Blackwellization)
- State of the art: Handling of more than 100 objects in real time

# Programs as Probability Models

- Probabilistic programming is a programming paradigm in which probabilistic models are specified and inference for these models is performed automatically.
- It represents an attempt to unify probabilistic modeling and traditional general purpose programming in order to make the former easier and more widely applicable.
- Probabilistic programming languages (PPLs) are usually based on traditional programming languages (e.g. Python, Matlab, Java etc.) and inherit their expressive power.
- https://en.wikipedia.org/wiki/Probabilistic_programming (contains also a list of PPLs)