Rao-Blackwellised Particle Filtering

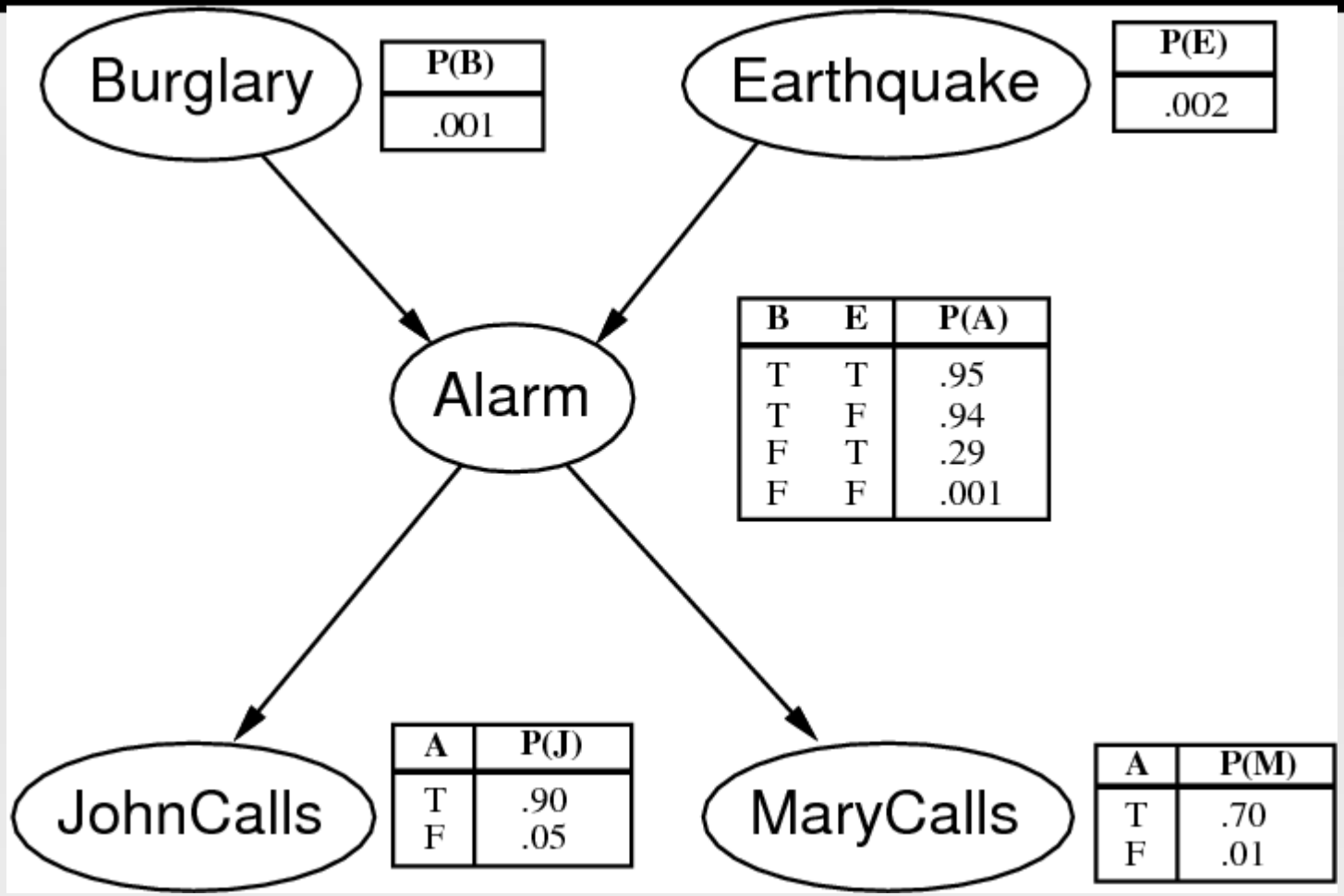
- Based on *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks* by Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russel
- Other sources
  - Artificial Intelligence: A Modern Approach by Stuart Russel and Peter Norvig

Presented by Boris Lipchin

Introduction

- PF applications (localization, SLAM, etc)
- Draw-backs/Benefits
- Bayes Nets
- Dynamic Bayes Nets
- Particle Filtering
- Rao-Blackwell PF

Bayes Net Example



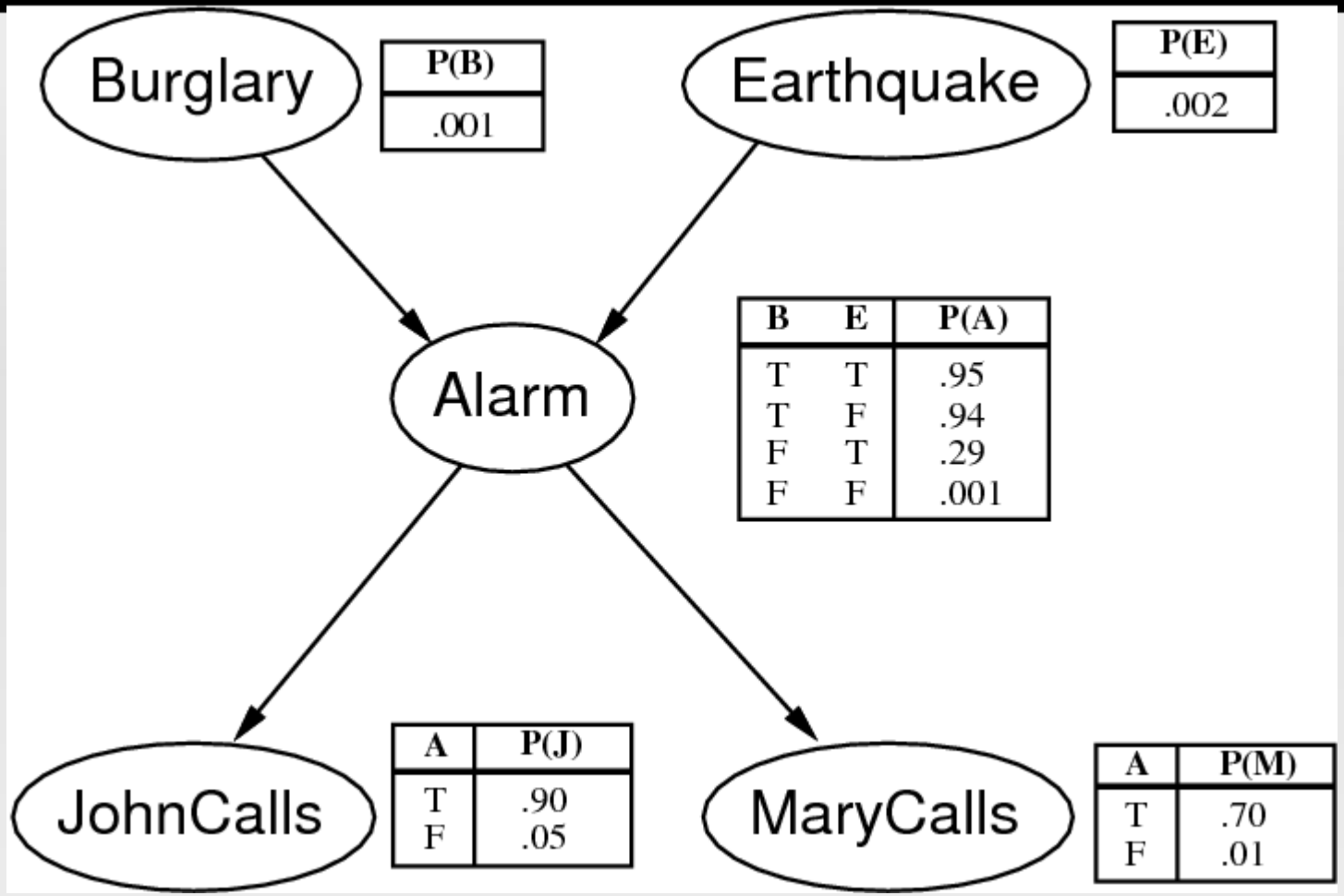
What is the probability of Burgalry given that John calls but mary doesn't call?

Adapted from Artificial Intelligence: A Modern Approach (Norvig and Russell)

Bayesian Network

- Digraph where edges represent conditional probabilities
- If A is the parent of B, B is said to be conditioned on A
- More compact representation than writing down full joint distribution table
- People rarely know absolute probability, but can predict conditional probabilities with great accuracy (i.e. doctors and symptoms)

Bayes Net Example



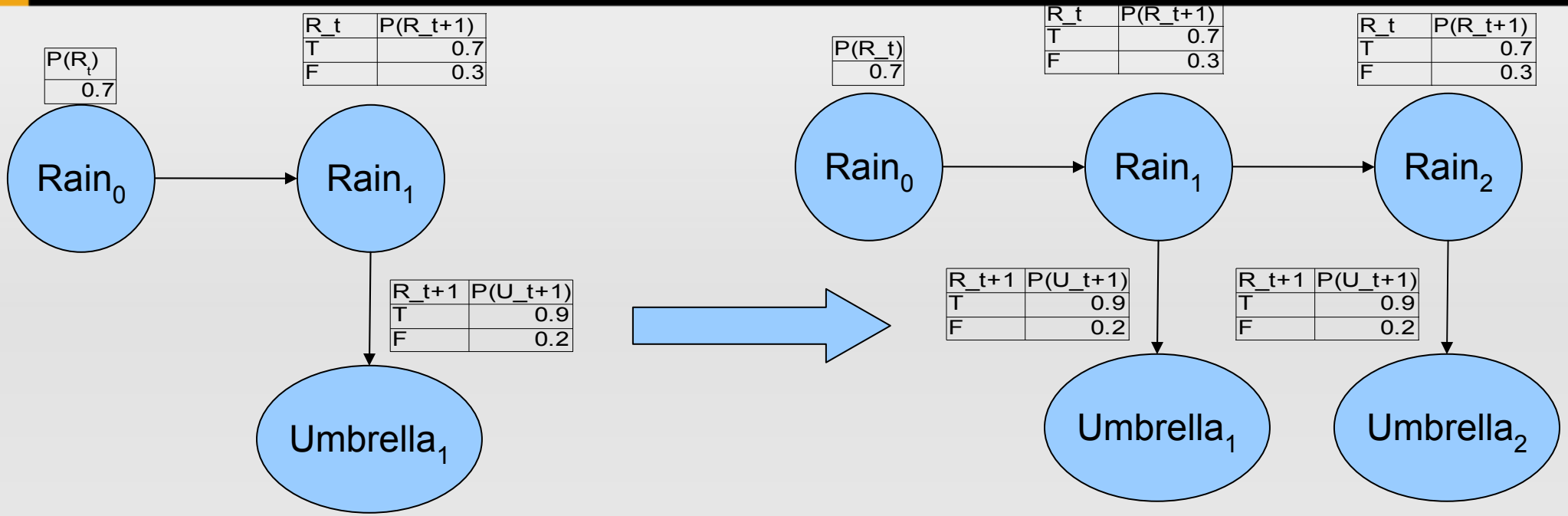
What is the probability of Burgalry given that John calls but mary doesn't call?

Adapted from Artificial Intelligence: A Modern Approach (Norvig and Russell)

Dynamic Bayesian Networks

- Represent progress of a system over time
- 1st Order Markov DBN: state variables can only depend on current and previous state
- DBNs represent temporal probability distributions
- Kalman Filter is a special case of a DBN
- Can model non-linearities (Kalman produces single multivariate Guassian)
- Untractable to analyze

Basic DBN Example



Adapted from Artificial Intelligence: A Modern Approach (Norvig and Russell)

DBN Analysis

- Unrolling makes DBNs just like Bayesian Network
- Online filtering algorithm: variable elimination
- As state grows, complexity of analysis per slice becomes exponential: $O(d^{n+1})$
- Necessity for approximate inference
  - Particle Filtering

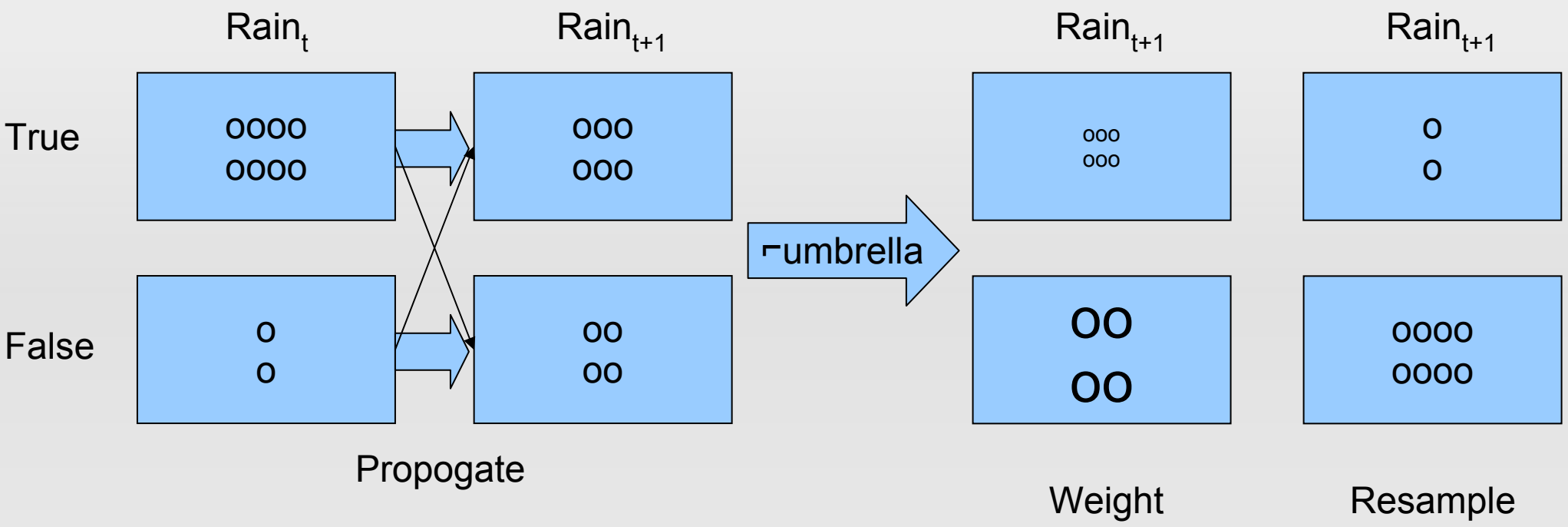
Particle Filtering

- Constant sample count per slice achieves constant processing time
- Samples represent state distribution
- But evidence variable (umbrella) never conditions future state (rain)!
- Weigh future population by evidence likelihood
- Applications: localization, SLAM

Particle Filtering: Basic Algorithm

- Create initial population:
  - Based on $P(X_0)$
- Update phase: propogate samples
  - Transition model: $P(x_{t+1} | x_t)$
- Weigh distribution with evidence likelihood
  - $W(x_{t+1} | e_{1:t+1}) = P(e_{t+1} | x_{t+1}) * N(x_{t+1} | e_{1:t})$
- Resample to re-create unweighted population of N samples based on created weighted distribution

Visual example



This method converges assymptotically to the real distribution as N → ∞

Adapted from Artificial Intelligence: A Modern Approach (Norvig and Russell)

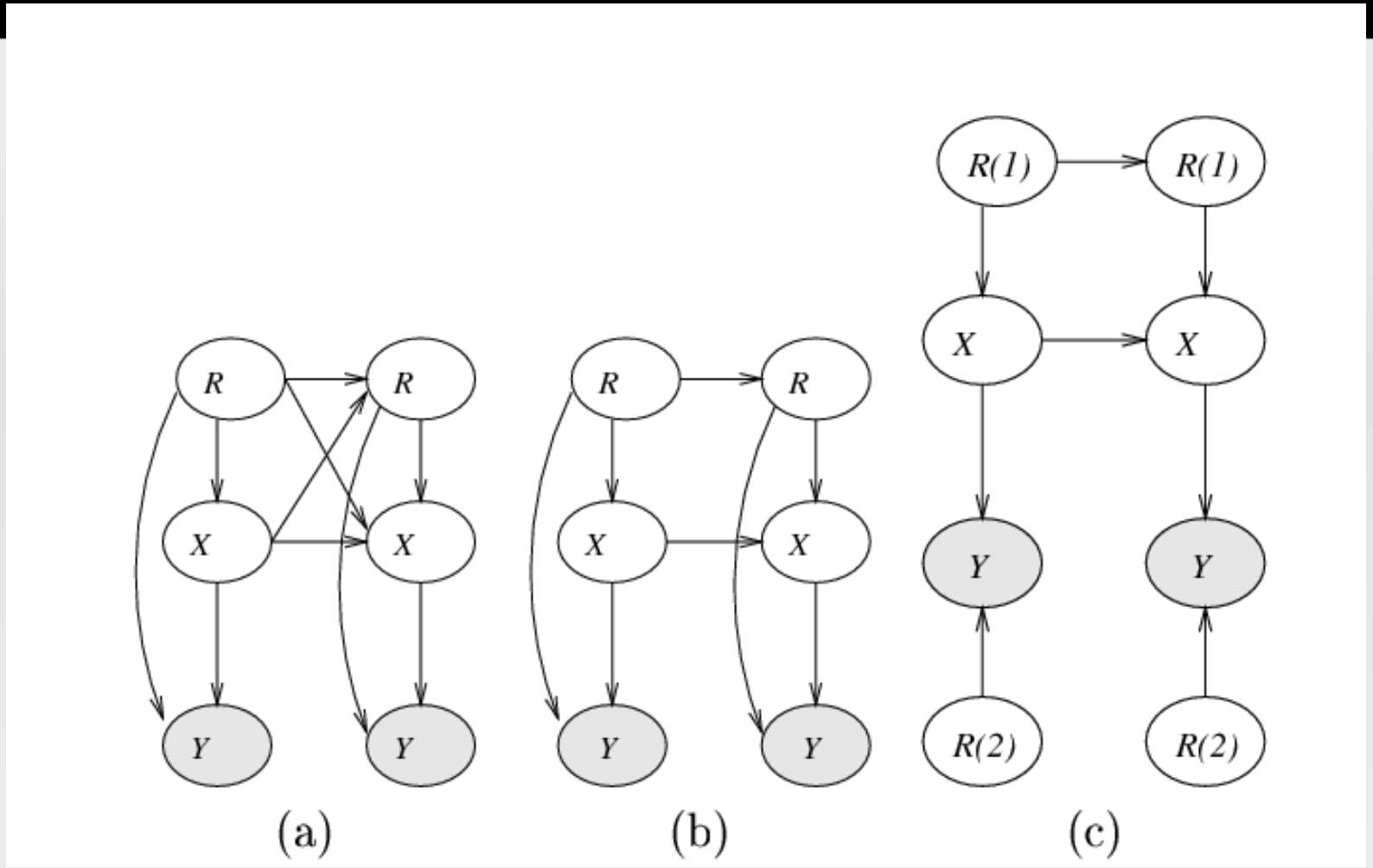
RBPF

- Key concept: decrease number of particles neccessary to achieve same accuracy with regular PF
- Requirement: Partition state nodes Z(t) into R(t) and X(t) *s.t.*:
  - $P(R_{1:t} | Y_{1:t})$ can be predicted with PF
  - $P(X_t | R_{1:t}, Y_{1:t})$ can be updated analytically/filtered
- Paper does not describe partitioning methods, efficient partitioning algorithms are assumed

RBPF: Concept Proof

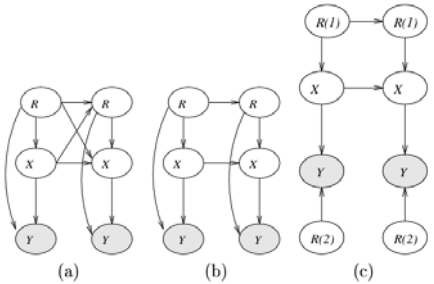
- PF approximates $P(Z_{1:t} | Y_{1:t}) = P(R_{1:t} , X_{1:t} | Y_{1:t})$
  - Remember state space Z partitioned into R and X
- $P(R_{1:t} , X_{1:t} | Y_{1:t}) = P(R_{1:t} | Y_{1:t}) * P(X_t | R_{1:t}, Y_{1:t})$
  - By chain rule property of probability
- Sampling just R requires fewer particles, decreasing complexity
- Sampling X becomes amortized constant time

RBPF DBNs – remember arrows



Adapted from *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks* (Murphy and Russell 2001)

RBPF DBNs



- R(t) is called a root, and X(t) a leaf of the DBN
- (a) Is a canonical DBN to which RBPF can be applied
- (b) R(t) is a more common partitioning as it simplifies the Particle Filtering of the root in the RBPF
- (c) Is a convenient partitioning when some root nodes model discontinuous state changes, and others some are the parent of the observation, and model observed outliers

RBPF: Algorithm

- Root marginal distribution: $P(r_{1:t}|y_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta(r_{1:t}^{(i)}, r_{1:t})$
  - δ is the Dirac delta function
  - w is the weight of the i-th particle at slice t and is computed by

$$w_t^{(i)} \propto \frac{p\left(y_t \middle| y_{1:t-1}, r_{1:t}^{(i)}\right) p\left(r_t^{(i)} \middle| r_{1:t-1}^{(i)}, y_{1:t-1}\right)}{q\left(r_t; r_{1:t-1}^{(i)}, y_{1:t}\right)}$$

    and then normalized.

- Leaf marginal:

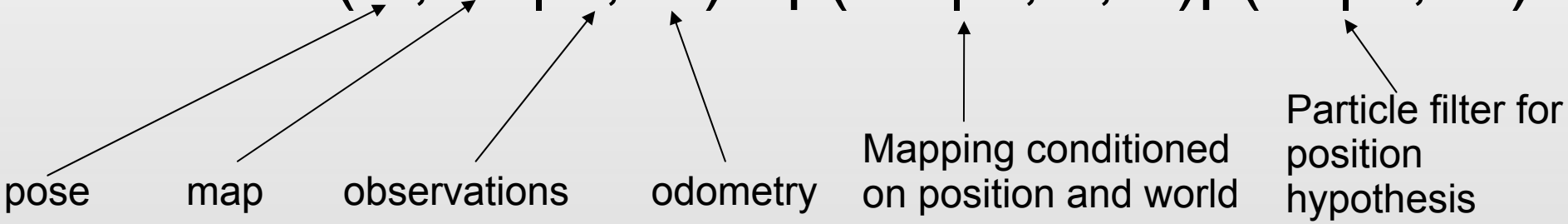
$$\begin{aligned} P(X_t|y_{1:t}) &= \sum_{r_{1:t}} P(X_t|r_{1:t}, y_{1:t}) P(r_{1:t}|y_{1:t}) \\ &\approx \sum_{r_{1:t}} P(X_t|r_{1:t}, y_{1:t}) \sum_{i=1}^{N} w_t^i \delta(r_{1:t}^{(i)}, r_{1:t}) \\ &= \sum_{i=1}^{N} w_t^i P(X_t|r_{1:t}^{(i)}, y_{1:t}) \end{aligned}$$

RBPF: Update, Propogate, Weigh

- The root particles in RBPF are propogated much like PF particles
- The leaf marginal propogation is computed with an optimal filter (Rao-Blackwellisation step)
- The leaf and root nodes together compose the entire state of the system, and thus can be weighted and resampled for the next slice.

Example: Localization

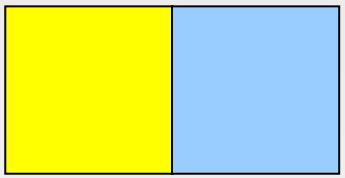
- SLAM: $P(x, m \mid z, u) = p(m \mid x, z, u)\, p(x \mid z, u)$



- m is the leaf, x is the root in the RBPF
- Particle updates based on input, expensive, we keep number of particles down

The Scenario

- Assume a world of two blue and yellow states labeled a and b (left to right)
- A robot can successfuly move between adjacent states with a probabilty P_m=.5 (transition model)
- The robot is equipped with a color sensor that correctly identifies color with probability P_c=.6



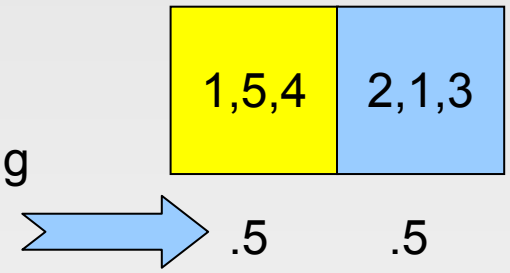
RBPF SLAM

- Using N = 5 particles
- P(X) represents state distribution (localization)
- P(M) represents color distribution (mapping)
- Prior for colors is an even distribution (unmapped)
- For simplicity, P(X=a) = 1, P(X=b) = 0

RBPF SLAM

- Randomly select particles according to prior distribution (labeled by number)
- arrow represents real robot position/detected color
- Create particles based on color

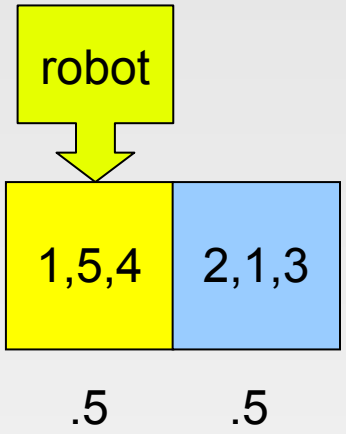
Represents mapping based on particle count



Remember: This means particle 1 hallucinates yellow in both boxes, particle 2 hallucinatees yellow only in right box and blue in left, so on and so forth.

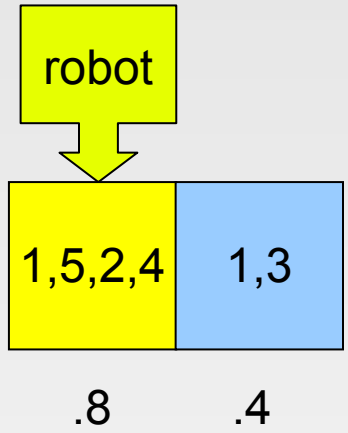
RBPF SLAM

- P(X(t)=a) = 1
- Calculate weights:
  - W1 -> P(E(a)=y | M(a)=y,M(b)=y)=
    P(E(a)=y)*P(M(a)=y | E(a)=y)*P(M(b)=y | E(a)=y,M(a)=y)/
    (P(M(a)=y)*P(M(b)=y | M(a)=y)) =
    .5 * .6 * .5 / (.5 * .5) = .6
  - W2 -> P(E(a)=y | M(a)=b,M(b)=y)=
    P(E(a)=y)*P(M(a)=b | E(a)=y)*P(M(b)=y | E(a)=y, M(a)=b)/
    (P(M(a)=b)*P(M(b)=y | M(a)=b))=
    .5 * .4 * .5 / (.5 * .5) = .4
  - You can calculate these guys ad nauseum



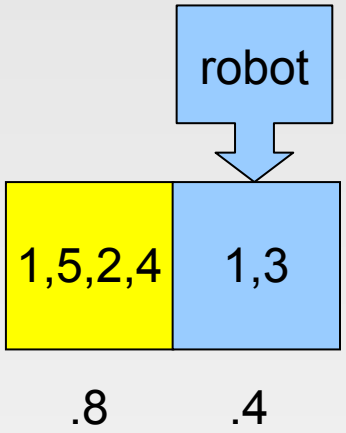
RBPF SLAM

- P(X(t)=a) = 1
- Next step is to resample based on weights (shown below)
- Find P(X(t)) distribution given previous state and current map



RBPF SLAM

- Calculate new weights
- Weigh samples and resample to obtain updated distribution for the particles
- estimate X(t) using optimal filter, evidence, and previous location



RBPF SLAM: Key Ideas

- Imagine if X(t) was part of state space
  - Calculations increase with number of states
  - Number of particles
- RBPF simplifies calculations by giving one a ”free” localization with an optimal filter

Questions?